Applications of Evolutionary Computation to Quadrupedal Animal Animation

James E. Murphy
B.Sc. (Hons.)

The thesis is submitted to University College Dublin

for the degree of Ph.D. in the

College of Engineering, Mathematical and Physical Sciences

March 2011

School of Computer Science and Informatics
Head of School
Prof. Joe Carthy

Research Supervisors
Dr. Michael O'Neill
Dr. Hamish Carr

Abstract

As humans are familiar with animal movement, a realistic animal animation must imitate this motion. This thesis explores how observations of natural evolution and evolutionary computation can be used to produce realistic quadrupedal animal animations, focusing on the grammar-based Genetic Programming method of Grammatical Evolution (GE).

A cross-discipline review of animation systems, biological knowledge and natural computing techniques applicable to animal animation is presented. Focusing on the horse, the construction of both a kinematic and physics-based model is described. The origins and representations of the data used to construct and animate these models are also discussed.

GE is applied to animation problems for the first time. Animating physics-based models is complex and a GE motion optimisation system successfully generates realistic, stable motions. Additionally, simple grammars with little domain knowledge generate novel movement.

For herd scenes, a GE-based system generates models of varying morphology and automatically optimises motion data for them. GE also evolves motion adjuster functions that dynamically modify limb movement based on the model's velocity. These functions are used in a real-time controllable kinematic animation system, in which a horse model moves with an accurate gait and executes gait transitions when necessary.

Overall, the use of GE and natural world observations is found to facilitate the generation of realistic quadrupedal animal animations.

Acknowledgements

I wish to thank my supervisors Dr. Michael O'Neill and Dr. Hamish Carr. I would also like to thank the members of the Natural Computing Research & Applications Group and the Imaging, Visualisation & Graphics Group for their help and input.

Additionally I would like to acknowledge my university examiners, Prof. Anthony Brabazon and Dr. Michela Bertolotto, my external examiner Prof. Simon Lucas and all others who have provided valuable contributions and opinions.

This research is co-funded by IBM under IRCSET's Enterprise Partnership Scheme and I am hugely grateful for this support.

Finally, I wish to thank my parents for their outstanding support throughout my time in university and Emily for her great patience and constant encouragement.

JAMES E. MURPHY

University College Dublin

March 2011

Contents

List of Figures

List of Tables

List of Equations

List of Grammars

List of Videos

The following is a list of links to the videos which accompany this thesis. In certain sections of the thesis, a link to the video corresponding to that section's content is provided. An online index of the videos is also provided at:

http://www.james-e-murphy.ie/thesis.html

Chapter 2 Animation:

2.1 Physics engine rigid body demonstration

2.2 Rag doll horse model

Chapter 5 Horse model:

5.1 Spring values set too low

5.2 Spring values set slightly too low or too high

5.3 Spring values set correctly

5.4 Spring values set too high

5.5 Spring values set too high (slow motion)

Chapter 6 Gait motion data and representations:

6.1 Animation from Muybridge photographs

6.2 Motion capture horse model

6.3 Published motion data (unoptimised)

Publications arising

1. James E. Murphy, Hamish Carr and Michael O'Neill. Grammatical Evolution for gait retargeting. In *Proceedings of the Eurographics UK Symposium on Theory & Practice of Computer Graphics, 2008. TPCG 2008*, pages 159-162, Manchester, England, June, 2008. Eurographics.

2. James E. Murphy, Michael O'Neill and Hamish Carr. Exploring Grammatical Evolution for horse gait optimisation. In *Proceedings of the 12th European Conference on Genetic Programming, 2009. EuroGP 2009*, pages 183-194, Tübingen, Germany, April, 2009. Springer.

3. James E. Murphy, Michael O'Neill and Hamish Carr. Gait optimisation for distinct horse models using Grammatical Evolution. In *Proceedings of the 15th International Conference on Soft Computing, 2009. MENDEL 2009*, pages 1-8, Brno, Czech Republic, June, 2009.

4. James E. Murphy, Hamish Carr and Michael O'Neill. Animating horse gaits and transitions. In *Proceedings of the Eurographics UK Symposium on Theory & Practice of Computer Graphics, 2010. TPCG 2010*, pages 215-222, Sheffield, England, September, 2010. Eurographics.

Chapter 1

Introduction

Quadruped animals such as dogs and horses are ubiquitous in our everyday lives and as such there is a large demand for animations of them in films, video games and advertisements. As humans, we are highly familiar with the motion of these animals both through direct interaction and from observing videos of their motion.

As such, if an animated animal does not exhibit motions equivalent to its real-life animal counterpart, this incongruity will be noticed by the viewers. In some situations, unrealistic stylised motion is deliberate and provides an effective break from reality. In other situations however, this lack of realism is unintentional and undesirable. In these situations, the viewers may consciously or unconsciously identify the inconsistency between the animated animal's motion and what would be observed in the real-world. The animation's illusion of reality is thus compromised and its overall effectiveness is diminished.

The objective of this thesis is to explore ways in which computer generated animations of quadrupedal, or four-legged, animals can be im-

proved in terms of realism, using Grammatical Evolution (GE).

We specifically focus on how the motion data used to animate quadruped models can be acquired. With this goal in mind, we investigate how observations of biology and natural evolution can be exploited to generate motion data using GE, an evolutionary computation technique. As such, the work presented in this thesis spans both the biology and computer animation fields. In addition to this, a third field known as natural computing, from which GE originates, is also explored.

Natural computing techniques comprise a wide range of human-designed computing systems inspired by natural processes. It would seem appropriate to examine the use of these systems for the computer generation of animal motion. We focus on evolutionary computation techniques, specifically GE, which simulate evolving populations of solutions over a number of generations, guided by a fitness function towards some goal.

These evolutionary computation approaches are inspired by the biological mechanisms of natural evolution. As the motion of an animal is the product of millions of years of evolution, it appears appropriate to apply these techniques to the problem of motion data production for animal animations.

In this thesis we thoroughly explore those aspects of biology that can be exploited for animation purposes. Focusing on the horse, several animation systems are described. As will be discussed, motion data pertaining to a real-life animal must be available if realism is an objective of an animation. As this motion data can be difficult to acquire, we explore different ways to use a small amount of data to produce a diverse range of animal animations.

For this purpose, we employ GE in a variety of experiments, the results of which are presented and discussed in this thesis. GE is chosen as it can evolve the structure of entire programs, rather than simply optimise a fixed set of parameters. Additionally, the grammar-based nature of GE means that complex problem domains can be easily represented through a grammar. The output phenotype structures are also human readable, which, if examined, may yield useful information about the problem domain.

In this chapter, our research aims are quantified into a set of thesis objectives and research questions in Section 1.1. In Section 1.2 the primary contributions of the thesis are summarised. Due to the multidisciplinary nature of the presented work, certain decisions must be made regarding scope and implementation; for each involved discipline, the corresponding decisions are summarised in Section 1.3. As a consequence of these decisions, the limitations of the thesis are described in Section 1.4. Supplementary to the primary contributions listed in Section 1.2, a list of secondary contributions including the data assembled and the developed applications and techniques are presented in Section 1.5. In the final section of this chapter, the structure of the thesis is outlined.

1.1 Research aims

The overall goal of the work presented in this thesis is to develop ways in which the realism of animal animations can be improved using GE. In order to clarify the aims of our research, it is important to define what we mean by realism in terms of animal animation.

In this thesis, an animal animation is considered realistic if it depicts an animal model whose morphology closely matches its real-life equivalent. The structure and proportions of the model are of utmost importance, whilst other aesthetic considerations are less of a concern.

In addition to structure, the model's motion must match that of a real-life animal. In terms of locomotion, the model must translate across a scene at an appropriate rate and that translation must appear to be caused by the interaction of the model's limbs with the ground. The sequence in which the limbs move, known as a gait pattern, must also be selected to correspond with the model's velocity.

At a lower level, the motion of the individual bones in each limb must move with respect to the real-life animal's physical constraints and be consistent with how an animal moves its bones for its various gaits.

1.1.1 Research questions

The ultimate goal of this thesis is to *explore how observations of natural evolution and evolutionary computation can be used to produce realistic quadrupedal animal animations, specifically focusing on the grammar-based Genetic Programming method of Grammatical Evolution.*

To address this goal, answers to the four research questions listed below are actively sought through research, development, experimentation and analysis.

1. *Can GE be used to optimise motion data for a physics-based quadruped model?*

2. *Can GE be used to retarget motion data from one physics-based quadruped model to another model of a different species?*

3. *Can GE be used to optimise motion data for multiple physics-based quadruped models of differing skeletal proportions for the animation of herd scenes?*

4. *Can GE be used to evolve functions that can dynamically adjust a kinematic model's motion based on its velocity and observations of natural locomotion?*

In response to these research questions, the main contributions of this thesis are summarised in the following section.

1.2 Primary contributions

Throughout the research process, many animation applications and systems have been developed and used for experimentation. Each major experimental component is separately published, as presented in the Publications Arising list on page xix.

The following are the primary contributions of this thesis.

Animation literature review

The development of quadrupedal animation systems and related techniques over the past 30 years is described in Section 2.4 with details given of seminal animation systems.

Optimisation and biologically inspired algorithms for gait generation literature review

The use of optimisation techniques and biologically inspired algorithms to generate gaits for biped and quadruped computer constructed models, and multi-legged robots is described in Section 2.4.2.

A number of contributions are made as a consequence of addressing the research questions listed in the previous section. These contributions are described below.

1. *Can GE be used to optimise motion data for a physics-based quadruped model?*

   (a) **Automatic physics-based motion data optimisation**

   A GE-based system for optimising motion data for a physics-based quadruped model is presented in Chapter 8 and is published [134].

   Of all the evolutionary computation methods, Genetic Algorithms (GA) are most commonly applied to motion data optimisation problems [73]. Our aim is to explore the use of Genetic Programming (GP), specifically GE, for the generation of motion data.

   A GP technique's ability to evolve the structure of a solution, rather than optimise a fixed set of parameters as in GA, deem it worthy of investigation for this class of problem. In addition, recent research indicates that GP techniques perform well if not better at certain motion generation problems [175].

The system presented in this thesis signifies the first time GE has been applied to an animation problem, specifically that of motion generation. In this system a GE implementation controls the evolutionary search while a physics-based quadruped simulation application acts as the fitness function.

Animating physics-based quadruped animal models is a challenging issue in computer animation. The motion data optimised by GE moves a physics-based quadruped model in a realistic and stable manner.

(b) **Measured motion data**

The issue of the inclusion of domain knowledge into the gait generation approach, in the form of motion data measured from a real-life animal, is also tackled in Chapter 8 and is published [134].

The question of domain knowledge inclusion and optimal AI ratio in GP is open [151]. We compare grammars that optimise motion data and grammars that are free to evolve novel motion. Some of the free-style grammars have no domain knowledge whatsoever and others have knowledge of an animal's muscles implicitly included in the grammar.

Those grammars that optimise motion data produce highly realistic, physics-based animal motions. The free-style grammars in comparison produce unique motions that allow the model to locomote at a high rate of speed, albeit without regard for the physical limits of the real-life animal. In the

grammars which implicitly include muscle information, many of the motions produced are comparable to that of a horse. A more sophisticated fitness function may further improve these results.

It is concluded that for realistic motion, domain knowledge must be included to some degree.

(c) **Rate of evolution**

In an evolutionary system, modularly varying goals can speed up the evolutionary process [98]. Inspired by this finding, we explore how the rate of an evolution can be affected through two experiments, presented in Chapter 8 and published [134]. We investigate whether generationally varying the weights of the fitness function components can replicate this speed-up phenomenon. In an expansion of this idea, the generational locking and unlocking of joints in the quadruped model is also explored.

Speed-up is only observed locally in the varying fitness function experiment and not at all in the joint locking experiment. In the latter experiment however, the number of valid, stable-motion yielding phenotypes produced from the earliest generations is greatly increased through the limiting of joint motion at the start of an evolution.

(d) **Terrain**

In the final experiment presented in Chapter 8, terrain traversal is briefly explored. The manner in which a model traverses

uneven terrain is another difficult and open issue for physics-based animal models. In response to this problem we employ a GE-based system to generate the motion required to traverse a simple terrain.

The result is impressive as an optimised motion allows the horse model to traverse a terrain at approximately twice the speed of a model whose motion is optimised for a flat running surface. This result demonstrates GE's potential to evolve more sophisticated terrain motion controllers for physics-based animal models and even robots.

2. *Can GE be used to retarget motion data from one physics-based quadruped model to another model of a different species?*

   Interspecies motion retargeting

   A GE-based system for retargeting motion data measured from one animal to an animal model of another species is presented in Section 9.1 and published [132].

   The retargeting of motions between characters often requires much adaptation and adjustment. This "motion retargeting problem", as it is dubbed, remains a topic of much research [71, 128, 83].

   The retargeting system presented in this thesis represents the first time an evolutionary computation technique has been applied to a quadruped model motion data retargeting problem. The system uses a GE implementation and quadruped simulation application to evolve motion data through a series of hybrid models towards a target animal model.

Both a discrete and continuous evolution system are described and experiments involving a horse to dog retargeting are presented. The retargeting attempts are found to be unsatisfactory mainly due to the large number of variables involved and inaccuracies of the joint-torque calculations in the hybrid models.

3. *Can GE be used to optimise motion data for multiple physics-based quadruped models of differing skeletal proportions for the animation of herd scenes?*

   Multiple models

   A GE-based system for generating motion data for multiple automatically generated quadruped models of differing skeletal proportions is presented in Section 9.2 and published [135].

   Using a GE implementation and a quadruped simulation application which generates models from allometric data and acts as a fitness function, physics-based horse models of differing proportions are generated based on an age parameter and animated using evolved motion data.

   Sequential and parallel approaches to model parameter and motion data optimisation are also explored. The sequential approach to model parameter optimisation is found to be best and GE is successful at generating motion data for multiple differing models in a herd scene.

4. *Can GE be used to evolve functions that can dynamically adjust a kinematic model's motion based on its velocity and observations of natural locomotion?*

   Kinematic gait and transition animation system

   Research is ongoing on the use of GP techniques for the evolution of controllers for various problems [2, 64]. A GE-based system for evolving functions which dynamically alter a model's motion pattern based on its velocity is presented in Chapter 10 and published [133].

   This is the first occasion that GE has been employed for the generation of dynamic motion controllers. These motion adjuster functions are evolved for use in a kinematic quadruped animation system which animates a model with gaits and transitions determined by a user-controlled velocity parameter.

   The GE-evolved dynamic adjusters are found to increase the realism of an animation by allowing a model's limb extents to reflect those observed in nature.

1.3 Research and implementation decisions

In tackling the research questions presented in the previous section, several additional implementation questions are raised regarding aspects of each of the three disciplines involved in this research.

Many of these questions (listed below) would be interesting research questions in their own right, however, for the purposes of this thesis,

decisions on technique and implementation are made based on findings in the literature and often short periods of experimentation.

1. **Animation**

   (a) *Which animation methods can be used for animal animation?*

   (b) *How can animal models be represented, constructed and animated?*

   (c) *What is the best source of motion data for animations?*

   (d) *How can motion data be intuitively represented for animation systems?*

   (e) *Can motion data be manually generated or optimised?*

2. **Biology**

   (a) *Which aspects of an animal's musculoskeletal system are most important when creating an animation?*

   (b) *What aspects of dynamic similarity theory can be exploited for animation purposes?*

3. **Natural computing**

   (a) *How can motion data be represented in a GE grammar?*

   (b) *What type of fitness function can be used to evaluate an animal model's motion?*

   (c) *Are multivariable fitness functions viable for motion generation problems?*

(d) *How does the size of the search space affect the quality of generated motion data?*

(e) *Would human involvement through an interactive fitness function assist a motion generation problem?*

(f) *Could use of a multi-objective evolutionary algorithm be of benefit for motion data optimisation?*

The questions raised here shall be discussed in the relevant sections of this thesis. The implementation decisions made limit the scope of the research and provide a definite problem domain in which evolutionary computation techniques can be explored. In the following section, the scope limitations are presented.

1.4 Limitations

The following subsections list the limitations of this thesis. As the research involves three distinct disciplines, some significant simplifications are required. These simplifications, limitations and implementation decisions are described below.

Animation

- The quadrupedal animation literature review focuses on animation systems that are not entirely reliant on motion capture yet aim to produce realistic motion.

- The principle goal of our animation systems is to produce realistic motion. Issues such as aesthetics of the model are a secondary

consideration. As such, matters such as surface deformations due to muscle motion are not addressed. Additionally, skinning techniques are briefly mentioned yet are not discussed in any detail.

- The animation techniques described in this thesis rely on Open Graphics Language (OpenGL) for the graphical rendering. An in-depth discussion on how three-dimensional computer graphics are created is not provided.

- Similarly, our use of the Open Dynamics Engine (ODE) is not described in any detail as the physics-based model construction process could be applicable to other physics engines.

- The physics-based horse model is essentially an open-loop model; the motion of its limbs and neck is computed without regard for environmental feedback. The balancing system however, utilises some feedback as the tilt of the model's trunk determines the minor correcting forces that are applied to the model.

- The terrain traversal experiments are limited to a single terrain as a proof of concept. Feedback loops and dynamic motion controllers are not utilised.

Biology

- Horse biology information is limited to that of direct use to animators. For example, knowledge and data pertaining to animal conformation and allometry is limited to that which can be used to determine the proportions of a computer constructed model.

- Dynamic similarity theory is fundamental to many of the presented experiments, however, related issues such as cost of transport and metabolic processes are not discussed.

- An introduction to genetic processes is provided in Appendix Section A.1.1. The description of some of the processes is simplified and details of chemical composition and variation are not included.

Natural computing

- Low-level details such as choice of evolutionary search parameters are not addressed in this thesis. A set of general parameters are chosen based on the literature and recommendations from other GE practitioners.

- In the gait generation and optimisation literature review, the robotics papers mentioned present research which is of particular interest to our motion generation experiments. Motion optimisation is a large topic of research in the robotics field and GAs are the most popularly used evolutionary algorithm. Rather than provide an in-depth review of robotics research, the referenced papers highlight the variety of optimisation techniques applied to the motion optimisation problem.

1.5 Secondary contributions

The experimental systems described in the Primary contributions section comprise multiple animation applications. While these applications of-

ten act as fitness functions within an optimisation system, they are also standalone programs capable of producing animations, and therefore significant contributions in their own right.

The experimental systems also exploit knowledge and techniques from the biology and animation fields respectively. The assembly of this biological knowledge and development of these animation techniques are also considered to be significant contributions and are included in the following list of secondary contributions of this thesis.

1.5.1 Background research

Equine information and data

Detailed information is assembled pertaining to horse model construction and animation. The structure and motion of the equine musculoskeletal system is described in great detail in Section 4.3 and a simplified horse skeleton which can be directly used to create an articulated model is presented in Section 5.1.2. The natural gaits of the horse are also described in detail in Section 4.4.

1.5.2 Applications developed

Curve Modifier Application (CMA)

An application designed to aid the manual and automatic kinematic motion generation process is presented in Section 7.1. The application provides visualisations and dynamic similarity-based motion scoring to evaluate motion data.

Motion Data Development Environment (MDDE)

An application designed to aid the manual physics-based motion generation process is presented in Section 7.2. This application simulates and animates a quadruped model and provides the user with a rich selection of scene navigation controls and adjustable simulation and model parameters. Motion data can be edited and evaluated in real-time using the configurable, interactive visualisations.

Physics-based Quadruped Simulation (PQS)

An application which simulates a physics-based quadruped model in motion and scores input motion data based on gait characteristic predictions and energy efficiency is presented in Section 8.1.2.

Variable Morphologies System (VMS)

An extension of the PQS called the VMS is described in Section 9.2.2. The VMS takes allometric data as input and can calculate, construct and animate a physics-based quadruped model with body proportions determined by a user-specified age.

Kinematic Gait Transition System (KGTS)

A kinematic animation system that animates a quadruped model using the correct gaits and transitions for that model's velocity is described in Section 10.3. The system dynamically modifies the supplied motion data to exhibit highly realistic motion based on a velocity parameter specified by a hardware controller.

1.5.3 Animation techniques and data representations

Kinematic horse model construction

A method for constructing and animating a hierarchical kinematic horse model with OpenGL is presented in Section 5.2.

Physics-based horse model construction

A method for constructing and animating a physics-based horse model with OpenGL and ODE is presented in Section 5.3 and further detail is provided in Appendix Section B.1.4.

Horse model file format

A bespoke file format which contains the skeletal and positioning data from which the kinematic and physics-based horse models can be constructed is presented in Appendix Section B.1.2.

Sinusoidal motion data representation

A motion data representation based on a summation of sinusoids is presented in Section 6.2.1. A Fourier decomposition and simplification process for converting discrete value motion data to this representation is also presented.

Piecewise motion data representation

A motion data representation based on a sequential description of curve segments is presented in Section 6.2.2.

Each of these secondary contributions are referred to and employed throughout the thesis, the structure of which is described next.

1.6 Structure of thesis

This thesis is divided into four parts, not including this chapter, which present related work, construction and animation techniques, experiments and conclusions respectively.

Part I: Related work

In Part I, all of the related work for the thesis is presented.

This part is divided into three chapters; animation, natural computing and biology. The animation related work chapter introduces kinematic and physics-based animation techniques. Also provided is a review of the animation literature, in which seminal animation systems are described.

In Chapter 3, aspects of natural computing are discussed. Biologically inspired algorithms and evolutionary computation methods are introduced followed by a detailed introduction to Genetic Algorithms, Genetic Programming and Grammatical Evolution. This chapter frequently refers to natural genetic processes and for readers unfamiliar with genetic terminology, there is a detailed introduction to genetics provided in Appendix Section A.1.

The final chapter of Part I discusses some biological knowledge that can be exploited for the creation of realistic animal animations. Although the research presented in this thesis is generally applicable to quadrupedal animal models, the experiments and data relate to the horse.

The chapter begins with a description of equine evolution, breeding and conformation. Following on from this, relevant aspects of the horse's

anatomy are discussed and then the musculoskeletal system is described in detail. The movement of the horse, in particular its gaits and transitions, are then discussed followed by a final section on dynamic similarity.

This chapter contains highly detailed information regarding horse morphology and behaviour which is employed for animation purposes at points throughout the thesis. The chapter itself can therefore be viewed as an information source for the production of realistic animal animations.

Part II: Model creation and manual gait generation

In this part, the computer construction and animation of horse models is described.

In the first chapter, the origins of data that can be used for model construction are discussed. Two different horse model construction and animation methods are then described; kinematic and physics-based. In Chapter 6, the gait motion data which is used to animate these models is described in terms of its origins and representations.

In the final chapter of Part II, the manual adjustment of motion data is described. Two motion data development applications are presented; one for kinematic animations and the other for physics-based. This chapter concludes with a description of how the physics-based horse model described in Chapter 5 is animated using the gait motion data discussed in Chapter 6, after a laborious manual motion data and model parameter tuning process.

This manual experience motivates the need for the automatic motion data optimisation solutions presented in the following part.

Part III: Experiments with GE

Part III comprises three chapters detailing the automatic motion optimisation systems and experimental findings summarised previously in Section 1.2.

Part IV: Conclusions

In the final part of this thesis, the work presented is summarised and conclusions are drawn. The research- and implementation-questions raised in Chapter 1 are briefly answered based on the findings and conclusions made throughout the previous parts. In the final section, the thesis concludes with a discussion of potential future research directions.

Note: a glossary of biological terms is provided immediately following this final part, starting on page 313.

Part I

Related work

In Part I of this thesis, three distinct areas of research are separately introduced, namely animation, natural computing and biology. Knowledge and techniques from each of these fields are involved in our exploration of how evolutionary observations and computation techniques can be exploited for the production of realistic quadrupedal animations.

In Chapter 2, computer animation techniques are discussed and some influential animal animation systems are described, including those which utilise evolutionary-based optimisation methods.

Following on from this, Chapter 3 provides an introduction to several popular evolutionary computation techniques and explains why Grammatical Evolution is chosen for the experiments presented in this thesis.

Finally, Chapter 4 describes areas of biology that can be exploited when creating realistic animal animations. The chapter includes a description of equine evolution, anatomy, motion and introduces dynamic similarity theory which plays a crucial role in the experiments presented later in the thesis.

Chapter 2

Animation

Animation is the rapid display of a sequence of still images to create the illusion of motion [186].

Animation techniques exploit the eye's persistence of vision phenomenon, in which an afterimage persists on the retina for a short period of time. This enables an observer to interpret a sequence of still images as if they were moving [155]. This is the fundamental concept of animation.

In this chapter, two classifications of animation, kinematic and physics-based, are separately described. Following this is a discussion on the development of animal animation approaches over the past three decades, focusing on a few seminal animation systems. The use of various optimisation and evolutionary computation techniques for generating animated character and robot motion is then examined. In the final section of this chapter, conclusions are drawn as to which animation methods and motion generation strategies are most suitable for quadrupedal animation.

In advance of this however, the following section presents a brief overview of the history and applications of animation.

2.1 History and applications

The earliest animation techniques, dating from the 1800s, involved the use of physical devices, in which some mechanism flipped between two or more still images. In the early twentieth century, what is now known as "conventional animation" was invented, where two-dimensional, hand-drawn images were filmed individually and then displayed in rapid succession [155]. This hand-drawn approach to animation dominated most of the twentieth century, helped along by advances in technique and technology.

The first attempts at computer animation appeared in the late 1960s and early 1970s. By the 1980s, there were a growing number of commercial, computer-based animation projects and companies. Since the mid-1980s, high quality computer animations have been regularly incorporated into live-action films, television and advertising [109].

In 1995, Pixar's *Toy Story* became the first full-length feature film to be entirely made using computer generated 3D animation. Since then, multiple big-budget, high-grossing computer animated films have been released every year. Most of these films contain a large cast of animated characters, each of which is created with great attention to detail as the success of a film is largely dependent on the appeal of the characters to the audience [112].

Animated characters are also prolific in modern video games. While the earliest games created in the 1950s and 1960s utilised extremely simple graphics, often consisting of single points in motion, many modern games feature fully articulated, aesthetically realistic characters.

As stated previously, this thesis is specifically concerned with the computer-based animation of animals. Rather than focus on the aesthetic appearance of these animated models however, the presented work centres on the production of realistic animal motion.

There have been many different approaches to animal animation proposed and employed since the earliest computer animated characters. Most of these approaches can be broadly classified as being either kinematic or physics-based techniques. In the following section, kinematic animation and the hierarchical kinematic model are described.

2.2 Kinematic animation

An animation method is described as being kinematic if the motion of the objects or characters in a scene is calculated without reference to the forces that cause that motion [155].

Figure 2.1: Popular characters from films by Pixar. (Images © Pixar)

Kinematic animation techniques allow animators to construct and move a character in any manner they wish, ignoring physical laws and real-world constraints if need be. Considering the enormous popularity of Pixar's cartoonish characters (see Figure 2.1), it can be inferred that audience appeal is often more important than rigorous physical realism, and a kinematic approach to animation can accommodate this.

A basic kinematic computer animation system has an artist create a model and direct that model's motion along a timeline. Rather than pose the model for every frame of an animation however, the animator often only sets the model's position and pose at strategic points in time called keyframes. The animation system then calculates the intermediate frames by interpolating between the keyframes, producing the full animation. This approach can approximate physical realism depending on the artist's aspiration and ability, or may deviate from reality entirely if so required [112].

When realism is necessary, a technique known as rotoscoping is often employed. Developed in the early 1900s, during the rotoscoping process an animator produces individual animation frames by tracing over the corresponding frames of a live-action film. In a traditional rotoscoping system, the individual frames of a film are projected onto a glass panel. A sheet of translucent material is placed on top of this panel and the animator then traces the projected image onto that sheet. The rotoscoping process is significantly simplified and automated by modern computer-based systems although the basic concept remains the same. The practice of rotoscoping is still in use today as the lifelike resultant motion can be difficult to achieve through unassisted hand animation [124].

To further improve the realism of an animation, motion capture techniques are often used. An actor's movements are first recorded using multiple cameras and position markers. That captured motion is then used to drive a character animation [78]. While this animation method produces highly realistic results, with subtleties that would be difficult for an animator to reproduce, there are drawbacks. The setup and mo-

tion capture process is expensive and some required motions could be impossible to act out.

The majority of modern character animations in movies and games use either motion capture, keyframe animation (often using rotoscoping) or a combination of the two. Motion capture is very effective when realism is required and the associated cost or scenario is not prohibitive. Keyframe animation, although less realistic, avoids the expense of actors and equipment and also allows an artist to produce motions that would be difficult or impossible to perform.

In this thesis, the focus is on realistic animal animation, particularly that of quadrupeds. Attempting to motion capture non-human animal movement may be made more difficult by unruly subjects and larger scale equipment requirements for bigger animals. As such, in the following section the focus is on non-motion capture, kinematic character animation techniques using a structured model.

2.2.1 Hierarchical kinematic modelling

An artist is usually more concerned with the overall form of a character's motion rather than with the frame by frame positioning of every movable body segment. A structured model approach such as hierarchical kinematic modelling is often employed, in which connectivity constraints are enforced among objects organised in a treelike structure [155].

Hierarchical models frequently comprise a set of objects connected end-to-end, forming multibody jointed chains. Animals are often modelled in this fashion as the joints between bodies can be manipulated to produce movement. The figure's connectivity is built into the model's

structure, and motion in one body will affect all other bodies farther down the chain. This property reduces the burden on an animator, as he or she need not be concerned with keeping moving bodies attached to one another.

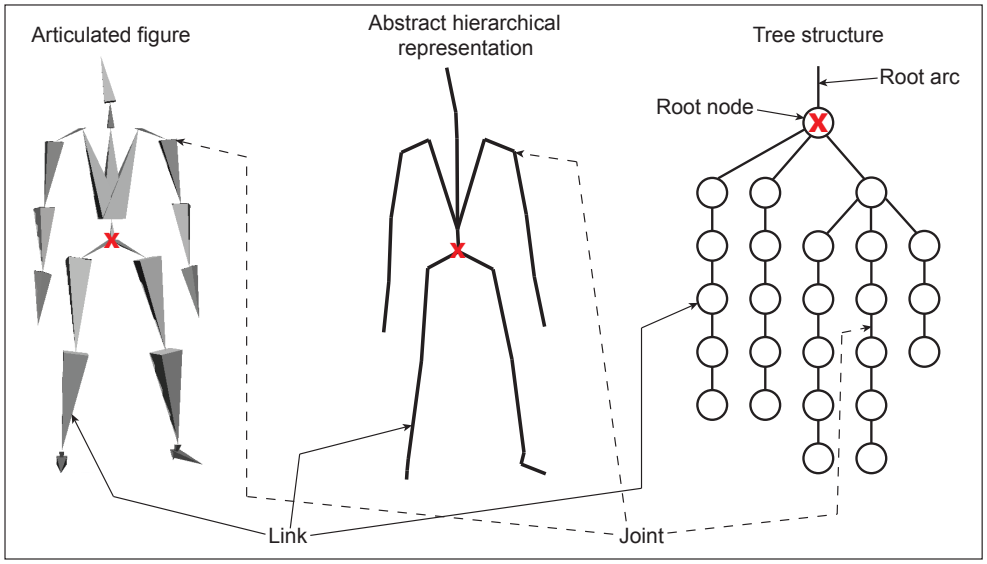


Figure 2.2: Illustration of how an articulated figure can be represented as a tree structure.

Figure 2.2 shows an example of a hierarchical model. The articulated figure is a simplified model of some real-life animal; a human in this case. The figure is composed of body segments and the joints between them, which can also be depicted as an abstract hierarchical representation comprising links and joints, as shown in the figure. By examining a model such as this, the link and joint objects can be organised into a treelike structure.

In this structure, a model is represented as a series of hierarchical linkages, illustrated as a set of nodes connected by arcs. The position indicated by the red cross on the articulated figure (and corresponding abstract model) becomes the root node in the tree representation. The

root node is the highest node in the tree and its position is known in global coordinates, with respect to some global origin. The position of all other nodes in the tree are relative to this root node. The closer a node is to the root, the "higher up" it is considered to be in the hierarchy.

Each node in the tree relates to and holds information about a particular link or body part. Each arc in the tree relates to a joint and contains the transformation (rotation and translation) that is applied to all nodes below it in the hierarchy. In Figure 2.2, the root arc represents a global transformation which when applied to the root node repositions the entire model in the global coordinate system.

In computer animation, the position and orientation of a model's body segments at each frame is specified by the set of transformations applied to each joint in the hierarchical model. Figure 2.3 shows an example of an articulated human figure, represented as a hierarchical model, performing a kicking motion. For each frame of the animation, rotation values for each joint in the model are supplied. Using this data, a sequence of transformations are applied according to the hierarchical model. The links in the model are rotated by some amount, with links lower down in the hierarchy being equally effected by transformations applied to links higher up in the hierarchy. In this manner, a kinematic animation is produced.

To be more specific, this method of animation is known as forward kinematics; angular information is supplied for each joint in a chain of bodies and the position of the leaf node, or end-effector, is calculated from this.

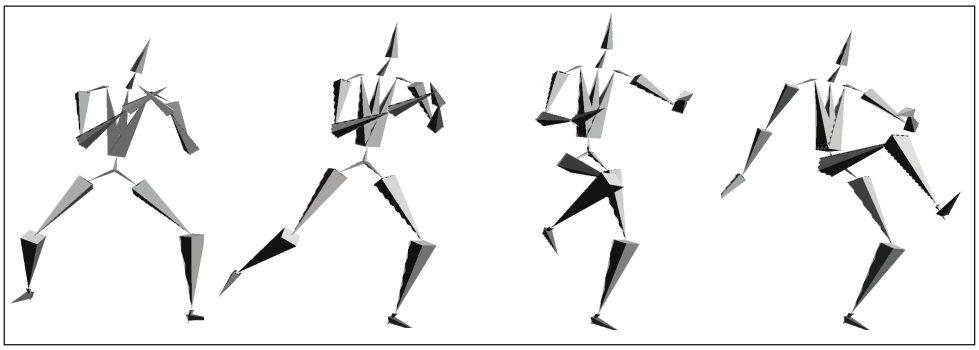


Figure 2.3: Still frames of a human kicking motion.

With another approach known as inverse kinematics, the artist only specifies the position that a link, such as a hand or foot, should move to; no joint-angle information is provided. It is left to the animation system to calculate by how much each joint in the chain should rotate in order to move this link to the required position. Although this appears to be an attractive solution, the major issue with inverse kinematics is that each problem can have zero, one or multiple solutions depending on how a system is constrained. If a solution is found, there is no guarantee that it will be aesthetically agreeable or physically realistic.

As realism is a goal of this thesis, the focus is on producing forward kinematic animations of animals and generating the sets of joint rotations that can move a hierarchically structured model in a lifelike manner.

As mentioned previously, a kinematic model can be moved without regard for dynamics. While an artist may attempt to constrain a model to behave in a physically realistic manner, it is not a requirement of kinematic animation. In the following section a technique known as physics-based animation is described, which introduces physical laws into the animation system, thus enforcing physical realism on a scene.

2.3 Physics-based animation

In contrast to kinematic animation, physics-based animation systems use the laws of physics to determine the behaviour of a computer constructed scene or model.

A physics engine controls the physical simulation of rigid bodies in a scene through integration of their equations of motion and the resultant animations are physically realistic. This technique also removes the need for an artist to hand-animate the frames of motion, however, some of the artist's control over a scene is relinquished.

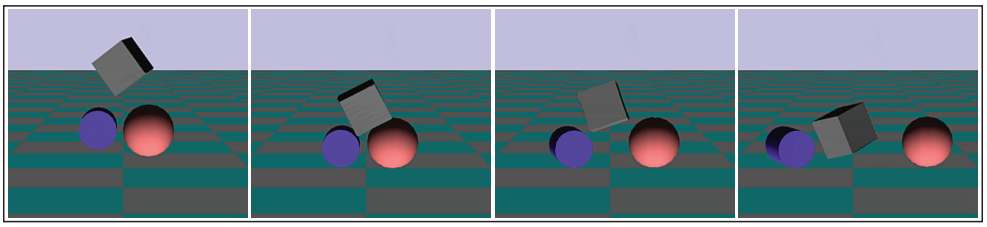


Figure 2.4: Physical interaction between three rigid bodies.
Video 2.1 Physics engine rigid body demonstration

The physics engine ensures that rigid bodies in a scene react in a physically realistic manner to gravity, friction, collisions, applied forces and torques [53]. In Figure 2.4, three rigid bodies fall in synchrony under the force of gravity. The collisions between the bodies and associated friction exert forces on each of them producing linear and angular accelerations. While this scene consists of a collection of unconnected objects, models comprising interconnected rigid bodies are also possible allowing for the creation of articulated animal models.

In a physics-based animal model, the rigid bodies can represent the animal's bones, or body segments, and the connections between these

bones are the joints. When connecting the rigid bodies, the degrees of freedom of each joint must be specified.

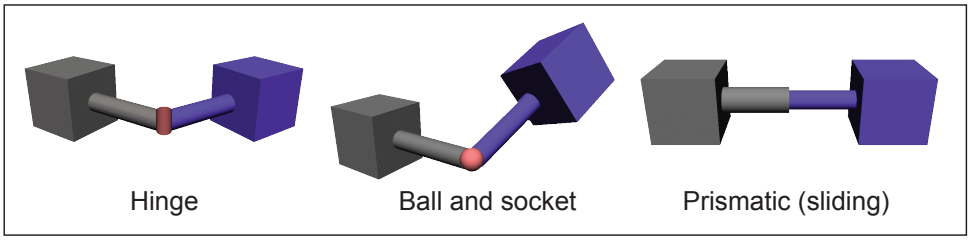


Figure 2.5: Three commonly used joints in computer animation.

The hinge joint in Figure 2.5 is an example of a simple joint with one degree of freedom allowing movement in a single direction. The ball and socket joint is an example of a complex joint with more than one degree of freedom. Both of these joints are the computer animation equivalent of biological joints (discussed in Section 4.2.2). The third joint in the figure is a prismatic or sliding joint that allows a single-axis sliding motion. While this type of joint is not usually found in nature, it can be used to approximate the shock absorption effect and elastic mechanisms observed in animal musculoskeletal systems.

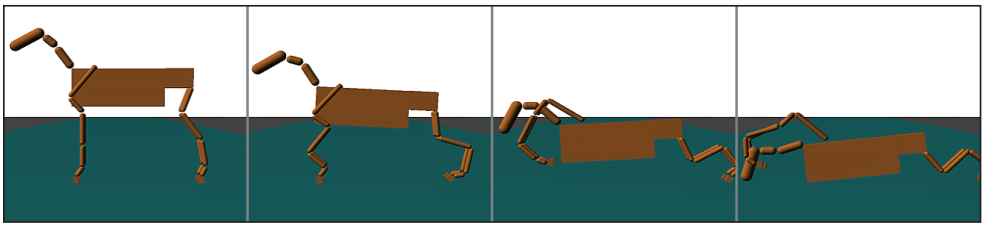


Figure 2.6: A physics-based horse model without constraints or torque on the joints. The model collapses to the ground under the force of gravity. *Video 2.2 Rag doll horse model*

The horse model shown in Figure 2.6 is constructed from a set of rigid bodies, each corresponding to a particular bone or body segment. Each

of these bodies is attached to one or two other bodies by an appropriate type of joint. In this example however, besides the degrees of freedom enforced by each joint's type, no additional constraints or torques are applied to the joints. The force of gravity therefore pulls the model to the ground like a rag doll, as can be seen in the sequence of images.

To produce motion in the model, or simply hold it up against gravity, torques can be applied about the joints, emulating muscle force and producing motion in the connected bones. As the bones are interconnected by joints, forces applied to a single rigid body may propagate throughout the rigid body system. For locomotion, a motion controller applies torques of specific magnitude about each of the joints in a limb with precise timing to produce the required sequence of bone rotations.

The high-level view of how motion can be produced in a physics-based articulated model presented above does not address the issue of how the torques on the joints are calculated. As with kinematic systems, physics-based animation approaches can be subdivided into inverse and forward subgroups.

In an inverse dynamics system, the desired end-position of a limb segment is known and the forces and torques required to move that segment to its destination are calculated backwards. Forward dynamics operate in the opposite fashion. Given the current state of the model, i.e. the position, forces and torques acting on its component bodies, the destination state of the model is predicted [53].

This description of physic-based animation is a highly simplified introduction to a nontrivial animation method. Further details of how a physics-based animal model is constructed are provided in Section 5.3.1.

Creating motion in the model and the issue of inverse versus forward dynamics will be revisited in Section 5.3.2.

While the construction of the scene or model is crucial to the success of the animation, the other fundamental component of a physics-based animation is the physics engine itself.

2.3.1 Open Dynamics Engine

The physics-based animation experiments presented in this thesis use the Open Dynamics Engine (ODE) [180], as it is well-regarded, stable and open-source. There are other physics engines available, but many are commercial products and are significantly expensive. ODE is an easy to use, open-source alternative and has been successfully applied to physics-based horse animation [119].

ODE is a high performance library that handles all of the low-level details of the physical simulation, including the physical motion of rigid bodies and collision detection with friction. The library is platform independent and its simple to use C/C++ API allows a user to construct articulated rigid body systems with relative ease, using the provided rigid body primitives and joint types.

ODE's supplied geometries include a sphere, box (cuboid), cylinder, capsule and user definable triangular mesh. When constructing a model or scene, these primitives are created with user-defined dimensions, mass, centre-of-mass position and inertia matrix. Each body is then given a starting position and orientation in the scene. Linear and angular velocities can also be set and external forces may be applied to the bodies during simulation.

If the user wishes to connect the rigid bodies together, several joint types are provided. These include hinge, ball and socket, prismatic (slider), universal and hinge-2. A universal joint is a more constrained version of the ball and socket joint and the hinge-2 behaves like two regular hinge joints in series. Each of these joints exhibit different behaviours and have user-adjustable parameters. One of the most useful adjustments is the ability to limit the degree to which an attached rigid body can rotate about one of the unconstrained axes of a joint.

Once the scene or model is constructed, the simulation can begin. The process of simulating a rigid body system through time is known as integration. At each integration step, the current time is advanced by a given step size and the state of each rigid body in the system is recalculated for that new time value. The entire scene can then be graphically displayed on some output device in its updated state. This continuous process of updating and displaying the state of the scene yields the animation. The ODE library itself does not produce any graphical output however, so it is the user's responsibility to use a graphics library to render objects corresponding to the ODE geometries, if so required. All of the animation examples in this thesis have been rendered using OpenGL [95, 176] which is discussed in Chapter 5.

There are many adjustable parameters in any physical simulation and ODE is no exception. Variables such as the joint stiffness, or the depth to which objects may interpenetrate can visibly affect the behaviour of objects in a scene. Other parameters such as the time-step size must be appropriately set in a trade-off between accuracy and performance of the simulation.

Stability is also an issue with this type of animation. A particular type of scene or model may be prone to "explosions" in which calculation errors accumulate and lead to non-physical behaviour. Problems such as this can be solved through adjustment of the simulation parameters or as a last resort, modification of the scene. This is one of the more common difficulties with creating a physics-based animation and will be discussed further in Section 5.3.3. A full description of the features and parameters of ODE can be found in the manual [180].

ODE summary

ODE is currently used in multiple computer games, simulation and 3D authoring tools [181]. The authors claim it is useful for simulating vehicles, virtual reality environments and virtual creatures. In this thesis, ODE is used to create a physics-based model of a horse, described in Section 5.3.

In the past, fully featured physics engines such as ODE and the computation power to run them were not available. As animal animation systems developed over the last twenty-five years however, elements of dynamics were incorporated and the resulting animations became increasingly realistic. This development of animal animation systems towards ever greater physical realism is discussed in the following section.

2.4 Development of quadrupedal animation

Quadrupedal locomotion is studied in both the computer graphics and robotics fields. In the computer graphics field, the goal of this research is to produce animations. In the robotics field however, analysis, simulation and animation of animal motion is often part of the robot development process.

Regardless of motivation, this mutual interest in how four-legged animals and robots move has produced a significant number of publications detailing observations of quadrupedal locomotion, animation systems and motion generation approaches.

The following section describes the development of quadrupedal animation techniques and related knowledge over the past three decades.

2.4.1 Animation techniques

The realistic animation of humans and other bipeds is a popular research topic [185]. The animation of quadrupeds is less well studied, however, animation techniques developed for use with biped models can often be applied to quadrupeds with certain modifications.

As previously stated, the realism of an animation created using a simple keyframe-based animation system is dependent on the skill and intent of the animator. A character's motion can be automated to some degree using a procedural approach, however, low-level control over the animation is often relinquished and the complexity of a character's articulation's can prove problematic.

The rotoscoping technique introduced in Section 2.2 has been extensively used for character animation throughout the last century. Disney often filmed real-life animals in order to rotoscope their motions [187]. This technique was used to produce the motion of quadruped characters in films such as *Bambi* and *The Jungle Book*.

Rotoscoping techniques can also be used to produce animal animations from a series of high-speed photographs of animal motion. The source of these photographs will be discussed in Chapter 6. Motion capture techniques, also introduced in Section 2.2, are often used for quadrupedal animation, however, the difficulties involved in capturing animal motion is often prohibitive. The subject of animal motion capture is revisited in Chapter 6.

In this section, the focus is on quadrupedal animation systems which do not rely entirely on motion capture data. Other alternative forms of motion capture (including statistical analysis [58, 68], contour tracking which allows for a more automated approach to rotoscoping [3] and automatic extraction of motion from video [202, 108]) are not discussed in detail.

From kinematic to physics-based animation systems

One of the earliest computer generated animal animation systems was introduced by Michael Girard and A. A. Maciejewski. In their 1985 paper [70], decades of mathematical, bioscience and robotics research are combined to describe a general model of legged locomotion. Focusing on legged animals in general, the animation is based on an artist-generated motion. This motion is described through input parameter values and

the model's feet are moved into a desired position by the artist. The joint-angles of each limb are then calculated using inverse kinematics. Whilst this is not a physics-based animation, the system ensures that accelerations of the model's body are synchronised with the timing of the forces that would be propagated through its legs, were it not a kinematic model.

In a later paper [122], McKenna states that accurate simulation of Newtonian mechanics is essential if a model is to move realistically. The paper describes how a cockroach is animated by numerical integration of its equations of motion. Locomotion is produced through the application of torques about its joints according to motion patterns based on biological observations.

Building on this work, Raibert describes how control algorithms are used in the animation of dynamic legged locomotion [165]. A biped robot, quadruped robot and kangaroo, which move with a specified limb pattern and speed, are modelled while control algorithms control the joints, model speed, direction and balance. The control algorithms translate a desired behaviour into control signals for the simulated actuators which in turn move the joints, producing movement.

It is worth noting that McKenna's [122] and Raibert's [165] multi-legged models have unarticulated limbs as can be seen in Figure 2.7. The joints in the hexapod's lower leg are rigid (**B**) and the biped and quadruped models (**C**) have limbs similar to pogo sticks. At that point in time, realistic physical modelling of an animal with articulated limbs such as a dog or horse was an open problem.

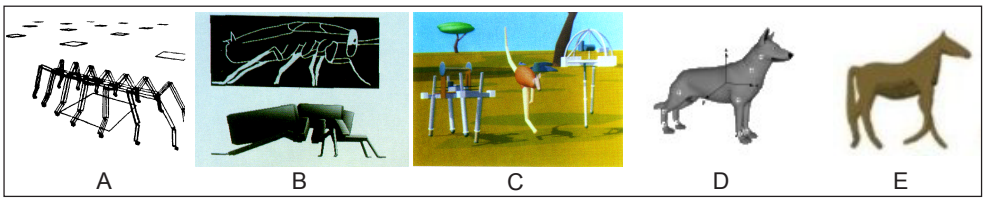


Figure 2.7: Images from important animation papers. **A.** Girard 1985 [70]: 14 legged insect. **B.** McKenna 1990 [122]: diagram of cockroach and hexapod model. **C.** Raibert 1991 [165]: quadruped, kangaroo and biped model. **D.** Kokkevis 1995 [102]: dog model. **E.** Marsland 2005 [119]: horse model.

Addressing this problem, Kokkevis presents a framework for animation and motion control of four-legged animals, specifically a dog [102]. The system uses a combination of kinematics and dynamics to produce movement; the legs are treated dynamically during ground contact and kinematically during the swing phase. A motion controller directs the stepping pattern based on limb pattern information extracted from photographs such as Muybridge's [136] (discussed in Section 6.1.1). Although Kokkevis's dog animations represent a step forward in terms of complexity, the dog's motion appears stiff and unnatural due to simplification of the model and constraints imposed on the freedom of movement of its joints.

Torkos produces animations of a 3D cat model using a combination of physics-based and kinematic animation [190, 191]. The presented approach is similar to the PODA animation system described by Girard ten years earlier [69]. Quadruped gaits are generated using constraints based on the location and sequence of the model's footprints. The motion of each limb's joints is calculated using inverse kinematics and dynamics calculations constrain the overall motion to appear realistic.

Some years later, Herr presents a two-dimensional numerical model of a horse at a trot and gallop, which again represent a growth in model complexity [88, 89]. Motion information is acquired from mechanical and energetic data published in the biomechanical literature while some of the movement data is also obtained from slow motion video footage of a horse in motion. The model correctly predicts that stride frequency and stride length change with speed.

Continuing with horses, Marsland and Lapeer describe a 3D real-time physics-based model of a trotting horse using the Open Dynamics Engine [119]. Motion data is extracted from a video of a horse using an active contour technique and morphological data is acquired from the biology literature.

Animation skeletons and meshes

The aforementioned papers each focus on producing motion in a model through manipulation of its animation skeleton. To produce an aesthetically realistic animation, an artist often places a 3D polygon mesh over this skeleton in a process known as skinning. The links of the articulated skeleton are attached to this mesh and as the skeleton moves, the mesh deforms accordingly. Issues such as skinning and the attachment process (rigging) are specific animation techniques and are beyond scope.

In a simple keyframe-based animation, the animator may simply manipulate the model's mesh into a sequence of poses and create the animation frames through interpolation of the keyframes. In many modern character animation systems however, the movement of the mesh is determined by the motion of an underlying skeleton.

The form and function of an animation skeleton is a highly researched topic and a subject which is discussed in a recent comprehensive review of quadrupedal animation methods and applications [179]. Assuming a skeleton is available, an animal's motion can be applied to that skeleton and the animation system will determine how the motion affects the overlying mesh. The motion of the skeleton itself is determined by the artist, rotoscoping, motion capture data or a procedural animation approach.

In situations when a skeleton is not available, given a particular mesh, it is possible to estimate where bones may be located by clustering the triangles of that mesh based on the transformation of the triangles [97]. Animations are often produced through direct deformation of the surface mesh without regard for the motion of an underlying skeleton. The manner in which the vertices of the mesh are moved varies between techniques. To afford a user useful control over the skeletonless mesh, certain limitations on the deformability of a mesh can be imposed based on the observation that adjacent vertices in a mesh often move together [48].

The triangle mesh concept can also be used to represent more than just the outer skin. The bones, muscle and other tissue of a real-life animal can be modelled using meshes, cylinders and ellipsoids; as bones move, muscles change shape and thus deform the surface mesh [204, 177].

Animation systems

The papers discussed previously in this section represent some of the more influential animal animation publications to date. In the game industry, most quadrupedal animations are created using a combination of hand-drawn, motion capture, inverse kinematic and procedural techniques.

The advantages and limitations of each approach determine the amount of control a user has over the animation and thus its suitability for a particular application.

In computer games, quadrupedal animations are commonly produced manually or through rotoscoping [179]. This can be time-consuming and unrealistic so to avoid the need for an artist to hand-animate individual frames of motion, several alternative techniques are also used. Inverse kinematics-based systems are commonly employed as well as procedural animation approaches; for example, a character's motion may be described as a sequence of body-part rotations over time.

Physics-based animations of biped and quadruped characters are relatively uncommon in practice due to the complexities involved in producing the models and creating stable animations. In those systems that do utilise physics-based characters, the animator may not be afforded low-level control over a character's motion; some systems allow the user to simply control the high-level behaviour of the character [113]. A typical behaviour-centric system has a user provide high-level behavioural commands to a model which are converted to the appropriate behaviour and thus animations by the system [27].

As well as allowing a user to directly control a model's behaviour, artificial intelligence techniques are also employed [189]. The motion and behaviour of a model depends on its morphology. Rather than have an artist create animations for every unique required motion, animation systems such as *Spore* create automatic motions and behaviours using procedural animation techniques [16]. The *Spore* system is based on a motion retargeting method which allows generalised data in a morphology-

independent form to be applied to specific characters using an inverse kinematics solver [83].

Robotics

As most of the earth's land is inaccessible to wheeled and tracked vehicles, legged robots that can tackle challenging terrain are valuable. Many of the motion, balance and foot placement techniques outlined in the robotics research papers are equally applicable to computer animation, and quadruped simulation systems are often described in the robotics literature [198].

The issue of uneven terrain is of particular interest in robotics [61, 54, 96]. Arguably the most advanced rough terrain quadruped robot in the world is currently Boston Dynamic's *BigDog* [164]. It is incredibly sophisticated, utilising an abundance of sensors to dynamically influence its stepping motion. In addition to *BigDog's* impressive array of onboard equipment, a rough terrain walking algorithm which was developed using physics-based simulations controls the robot's gait.

Also developed by Boston Dynamics is *LittleDog*; a quadruped robot designed specifically for locomotion research. *LittleDog* is used in many institutions for research into motor learning, dynamic control, rough terrain locomotion and perception. Sony's *AIBO* robot is another example of a relatively affordable robot which can be used for locomotion research purposes [93]. Many bespoke robots are also developed in research institutions for a range of locomotion challenges.

Robotics is a huge field of research and one which shall not be discussed here, however, some of the advances in robotic locomotion can

be equally applicable to quadrupedal animation. In particular the use of optimisation algorithms for robot gait generation has become commonplace and is of interest to us. Some examples of this are presented in the following section on motion generation.

2.4.2 Gait generation

A gait is a pattern of limb motion with which an animal (or robot) moves.

The animation papers described in the previous section use either artist-generated gaits or rely on motion data measured from real-life animals. In this section, the issue of automatic gait generation is explored.

An early method for generating character motions is presented in the seminal "spacetime constraints" paper [205]. In this system, a user inputs a model's physical structure and what resources that model has for creating motion; a model with limbs can push them against a ground surface to create movement for example. The user then specifies what the character is to do, and the manner in which it is to do it. This high-level description of the animation requirements, coupled with Newton's laws of motion, create a constrained optimisation problem. The paper describes a numerical optimisation approach which attempts to find a solution to this problem that is physically realistic, completes the task set and does it in the manner specified. This idea is subsequently built upon by allowing a user to interactively guide a numerical optimisation problem to an optimal solution [40].

Van de Panne presents another guided optimisation technique specifically designed to address balance issues [196]. In this physics-based animation system, balanced locomotion is learned in stages using a gradient

descent algorithm for the parameter optimisation. During the initial stages, balance is handled through the application of balancing forces. The influence of these stabilising forces is reduced in increments, as the model learns to balance by itself. The technique as presented relates to biped locomotion but is applicable to animals with other multiples of limb.

Another interesting animation technique is presented by van de Panne where the control mechanisms used for this physics-based animation technique are described as being analogous to wind-up toys and the models have no control over their balance [194]. It is instead assumed that the inherent stability of a gait is sufficient. The issue of stability is further explored where footprints are used as an optimisation constraint [195] and a numerical optimisation approach for improving stability in quadruped locomotion is presented [79].

One of the aims of Srinivasan and Ruina is to prove that human walking and running are the most efficient ways in which to move at particular velocities [182]. Using a minimal biped model, a numerical optimisation process discovers walking, running and a third energy efficient intermediate gait pattern. Other novel motion generation approaches rely less on the optimisation of data and more on creating smart motion control algorithms [21], often inspired by nature [87].

Fourier analysis of animal movement is also used for the production of gaits. The results of a Fourier analysis of periodic animal gaits are used as input to a simulated robot's joint actuator controllers [171]. Fourier analysis can also yield gait transitions for a computer-based legged animal model [126].

Most of the above mentioned papers use parameter optimisation approaches to produce balanced, physics-based motion. In the following subsection, a variety of papers that apply biologically inspired approaches to motion optimisation are introduced.

Biologically inspired optimisation examples

Biologically inspired algorithms (BIA) is a category of algorithms that mimic natural processes and are discussed in detail in Chapter 3. Popular BIAs include Artificial Neural Networks (ANN) and Particle Swarm Optimisation as well as Genetic Algorithms (GA) and Genetic Programming (GP) which are both classified as Evolutionary Algorithms (EA). BIAs are often used in robotics and animation systems for the automatic production of legged locomotion.

The *Euphoria* physics engine created by Natural Motion for example allows characters to be animated in a physically realistic manner whilst reacting in real-time to their environment and external forces [130]. The makers of *Euphoria* state that the engine simulates a computer constructed character's "motor nervous system, body and muscles" using Dynamic Motion Synthesis. Similar to the seminal work of Karl Sims [178], GA are used to evolve a form of Neural Networks as the controllers which determine the forces to apply to the skeleton [169].

The *Euphoria* system allows 3D characters to be interactively controlled and generates unique motions dynamically; this is in contrast to a system which exclusively uses predefined animation sequences (canned animation). The system has been used for biped characters in the American football game *BackBreaker* [129] to create unique tackles and to

animate horses in the Western game *Red Dead Redemption* [65]. The gait cycle motion of a character is based on heavily detailed motion capture and the *Euphoria* engine creates the physically realistic motion that occurs between motion capture sequences; these dynamic sequences usually occur due to external forces applied during gameplay.

Neural Networks are commonly used for gait pattern and transition generation [110, 115]. Examples include Jeff Clune's HyperNEAT-based system for generating quadruped gaits [39]. HyperNEAT is a generative encoding for evolving Neural Networks using the principles of the NeuroEvolution of Augmented Topologies (NEAT) algorithm. The HyperNEAT-based system is found to evolve impressive looking quadruped gaits with less human intervention than would normally be required when using a traditional EA to evolve a motion controller.

Gong et al. present a comprehensive review of evolutionary computation (EC) methods for gait optimisation [73] including applications of each of the aforementioned BIAs. Within this review, strong arguments are made in favour of using EC methods for gait optimisation problems. Focusing on specific EC approaches, the performance of GA and GP for quadruped robot gait generation has also been recently compared [175]. The performance of each EA is found to be dependent on the particular problem, however, GP, which is a far less popular EA than GA for gait optimisation, performed better in certain situations; a result which certainly warrants further investigation and is explored in this thesis.

GAs are the most commonly used EA for generating quadruped robot gaits [67, 206, 100]. Of particular interest is the work of Kiguchi et al. as the evolutionary search is concerned with finding a gait that uses

minimal energy, while covering a required distance at a specified speed [100]. This approach can be applied to gait generation for quadruped animal animations as they state that the optimal gaits produced for the robots are comparable to those expected of a real-life animal travelling at relatively similar speeds (see Section 4.5 on dynamic similarity).

While GA seems to be the gait generation EA of choice in robotics, other techniques are also used [73]. GP-based techniques for automatic quadruped robot gait generation are occasionally applied [174] and Particle Swarm Optimisation is found to generate fast quadruped gaits [172]. An autonomous, unspecified EA is used to generate gaits for the Sony *AIBO* Quadruped Robot [93].

The robotics papers discussed in this section represent a small portion of the literature available on the automatic optimisation of robot gaits. Although many different BIAs have been applied to this problem, the overwhelming majority of approaches employ some form of GA. This suggests that the suitability of GP for gait optimisation problems may be an open question and one which deserves investigation.

2.5 Chapter summary

In this Chapter, we have introduced the subject of animation and discussed two major animation classifications; kinematic and physics-based.

Physics-based animation techniques are very realistic and can automatically simulate highly complex interactions, given a well constructed scene or model. A major drawback of this approach however, is the nontrivial nature of the model construction and simulation setup. Sim-

ulations may also suffer from stability issues and for a computer game, the real-time computational cost of complex scenes can be prohibitive.

A kinematic animation technique in comparison suffers from none of the simulation related problems. In a computer game, as most of the motion information is produced by the artist in advance, kinematic animations incur relatively little computational expense. The downside of kinematic animations is that even highly skilled artists struggle to animate complex physical interactions. Further to this, it could be impossible to produce and store an appropriate animation for every possible physical interaction that could occur in a game. Both techniques have positive and negative aspects depending on the situation. In practice, often a combination of kinematic and physics-based animation is used.

In this thesis, both kinematic and physics-based animation of quadrupeds is explored. Building on the animation systems described in Section 2.4.1, both a kinematic and physics-based model of a horse is constructed and animated, as will be discussed in Chapter 5.

The motion data used to animate these models must be generated and optimised by some means, and there are several ways to do this, as was seen in Section 2.4.2. Numerical optimisation is a well established and regarded technique, however, the biologically inspired algorithms used so prolifically in the robotics field show great promise for animation purposes and warrant investigation.

In this thesis, we explore the use of evolutionary algorithms for quadrupedal animation. The field of natural computing is introduced in the next chapter and three evolutionary algorithms are presented and compared as candidates for quadrupedal gait generation.

Chapter 3

Natural computing

Natural computing is the study of computational systems inspired by nature. The term natural computing generally refers to three types of system: problem solving systems inspired by nature, computer systems that simulate natural phenomena and systems that use natural materials to perform computations [45]. In this thesis, the focus is on those systems that take inspiration from natural processes.

In the previous chapter, it was concluded that quadrupedal animation could benefit from the application of biologically inspired algorithms to the motion generation problem. Of all the biologically inspired algorithms available, those which are classified as evolutionary computation techniques seem particularly applicable, as animal gaits themselves are a product of natural evolution.

In this chapter, the large field of research known as evolutionary computation is described in Section 3.2 and the basic structure of an evolutionary algorithm is presented.

Section 3.3, 3.4 and 3.5 each provide complementary overviews of three popular evolutionary algorithms: Genetic Algorithms, Genetic Programming and Grammatical Evolution respectively.

This chapter concludes with a discussion on why Grammatical Evolution is specifically used for the experiments presented in the thesis.

The descriptions of evolutionary algorithms in this chapter frequently refer to genetic processes and terminology. A basic introduction to the aspects of genetics and the evolutionary process that will be referred to in this chapter is presented in Appendix Section A.1.

This chapter commences with a brief overview of the many biologically inspired algorithms in use today.

3.1 Biologically inspired algorithms

Biologically inspired algorithms (BIAs) are types of algorithms that mimic natural processes to solve some problem [30]. They have been successfully applied to difficult real-world problems as diverse as financial prediction[44, 30, 145] and model design [106, 1]. The most popular BIAs include evolutionary computation algorithms, swarm intelligence algorithms, Artificial Immune System algorithms and Artificial Neural Networks [45].

Inspired by the central nervous system, Artificial Neural Networks (ANN) are simplified models of the human brain created in silico or as a mathematical model. The basic structure of an ANN involves groups of artificial neurons which are interconnected by the computer equivalent of a synapse. An ANN is an adaptive system. During a learning phase,

the internal structure of the ANN changes based on information, both internal and external, that flows through the network. This learned structure of simple neural units and the connections between them can model complex input-output relationships and make ANNs highly applicable to pattern recognition problems.

As with ANNs, immunocomputing takes inspiration from animal physiology. The immune system is a highly complex system that must constantly recognise, destroy and remember both harmful foreign bodies and malfunctioning native cells. Artificial Immune System algorithms typically exploit the memory and learning ability exhibited by immune systems to solve complex pattern recognition problems.

Moving from internal to external behaviours, observations of the social interactions of organisms have been utilised in computational systems. Known as Swarm Intelligence, these systems typically comprise a self-organising population of simple agents who interact with each other and their environment. There is no central command centre, rather each agent follows a simple set of rules, with their interactions leading to emergent "intelligent" behaviour on a global scale. Comprising specific techniques such as Ant Colony Optimisation and Particle Swarm Optimisation, these systems are inspired by the workings of natural systems such as ant colonies, animal herding and bird flocking behaviour. Swarm intelligence systems have been directly used in the animation of animal flocks and crowd scenes in movies.

Ant Colony Optimisation systems are suited to path finding problems whilst Particle Swarm Optimisation techniques are robust to the difficulties caused by local minima in global minimisation problems.

As mentioned above, another major collection of BIAs are classified as evolutionary computation techniques and are discussed in the following section.

3.2 Evolutionary computation

Evolutionary computation (EC), a subfield of computational intelligence, is an umbrella term for a collection of BIAs adapting concepts of natural evolution and molecular biology [52].

EC techniques have been successfully applied to numerous engineering and design problems [106, 1] as well as computer games [64, 32, 2], musical composition [168, 201] and financial prediction [44, 30, 145]. When set up appropriately, these techniques can automatically solve problems that are nontrivial for humans. Results can be human-competitive and even improve upon the best human efforts [106, 127]. The reason that these EC algorithms are so adept at solving complex problems is the underlying evolutionary engine upon which they are based.

The environments that biological organisms inhabit are dynamic and highly complex. Through natural evolution these organisms adapt to outperform their peers in terms of survival and reproduction. An organism can increase its chances of being selected by being slightly better than the competition at attaining the available resources. It is this concept of "survival of the fittest" that is a cornerstone for each member of a set of EC techniques called Evolutionary Algorithms (EA).

For these EAs, the correlation between biology and computer science is reasonably straightforward. The environment in which an organism

lives is equivalent to a problem that is to be solved. It follows that the biological organism is analogous to a solution to that problem. The performance of a solution at solving a problem corresponds with the concept of biological fitness and the best performing solutions are consequently selected, as occurs in natural selection. Essentially, an EA simulates those evolutionary mechanisms described in Appendix Section A.1.2, namely selection, reproduction, recombination and mutation. The organisation of these mechanisms in an EA is the same as that of a natural evolution system.

3.2.1 Evolutionary algorithm overview

An EA solves combinatorial optimisation problems through the simulation of evolving populations of solutions which are guided towards an optimal solution by use of a fitness function.

The fitness function is always dependent on the problem and has to reliably measure the quality of a solution. In cases where the fitness of a solution is very difficult to define, such as the quality of a generated piece of art, an interactive fitness function may be used exploiting the human ability to evaluate the unquantifiable "goodness" of something. For best results, this fitness function must be carefully chosen in tandem with the phenotypic solution representation, whose variable characteristics will define its fitness performance.

Figure 3.1 presents the basic pseudocode of an EA, alongside an illustration of the natural evolutionary engine. An examination of this figure should elucidate the relationship between a natural evolutionary system and an EA. The major difference between the systems is the EA's very

specific start and end points, determined by the population initialisation step and termination clause respectively. Aside from this disparity, the EA process functions comparably to its natural counterpart.

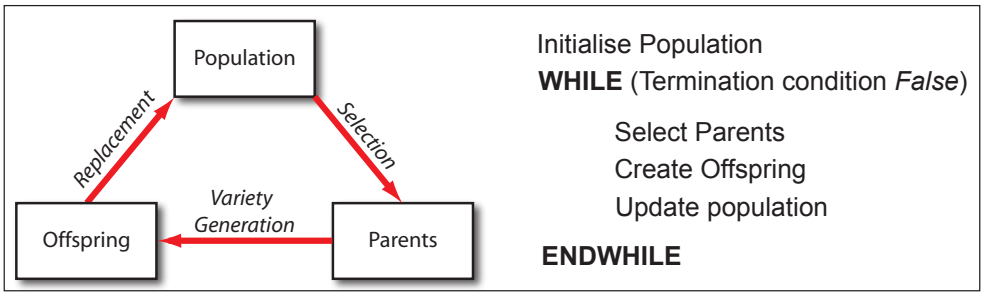


Figure 3.1: The cycle of natural evolution and the basic evolutionary algorithm pseudocode.

From an initial population, individuals with above average fitness are selected to reproduce. These parent individuals produce offspring, whilst genetic variation is provided through a recombination process inspired by chromosomal crossover. Further variation is introduced through mutations. These offspring individuals are then added into the population, replacing some previously resident individuals. The entire cycle continues until some termination condition is met.

There are many initialisation, selection, recombination, mutation and replacement techniques in use today. The specific choice of technique varies between EAs and are chosen to provide appropriate, balanced variety generation. Probably the most distinguishing feature of an EA is its genetic representation which encodes the physical structure and behaviour of its individuals. Details such as this will be discussed in relation to specific EAs in the upcoming sections.

There are five major metaheuristic optimisation algorithms that are classified as EAs, as shown in Figure 3.2. Genetic Algorithms [91, 127],

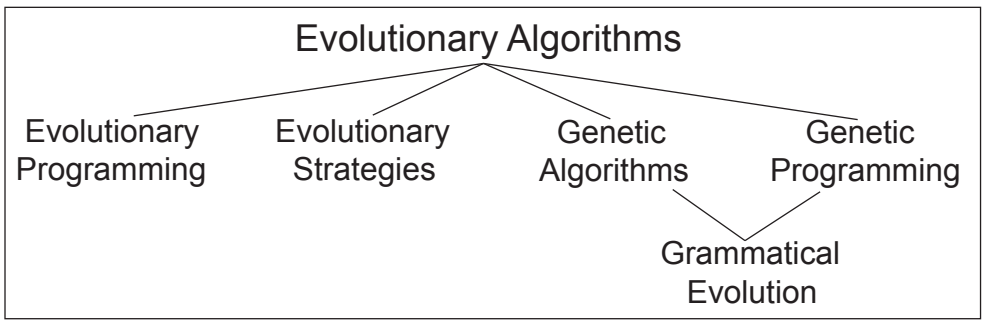


Figure 3.2: Evolutionary algorithms.

Genetic Programming [105, 22] and Grammatical Evolution [149, 150] will be explored in some detail in Sections 3.3, 3.4 and 3.5 respectively. Before this, Evolutionary Programming [60, 52] and Evolutionary Strategies [167, 26] are very briefly introduced.

3.2.2 Evolutionary Programming

Evolutionary Programming (EP) is a broad dialect of EC. Developed in the USA by Lawrence J. Fogel in the 1960s, EP was originally applied to the task of generating artificial intelligence using finite-state machines as predictors [60].

Some original variants of EP resembled modern Genetic Programming (see Section 3.4) except that in those early EP systems, solutions to a problem were found by evolving the parameters of a fixed program structure. The modern, broad definition of EP has no set genetic representation although the genotypes are often fixed-length character strings and mutation is the primary variation operator [52].

At its current point of development, EP is considered very similar to another EC technique, Evolutionary Strategies.

3.2.3 Evolutionary Strategies

Although similar to EP, Evolutionary Strategies (ES) was created separately in the 1960s, and was subsequently developed by Ingo Rechenberg and Hans-Paul Schwefel in Germany in the 1970s [167].

The ES representation is a vector of real numbers. As with EP, ES uses mutation and selection as the primary search operators. ES is distinguished however, by its set of mutation (self-adaptive rates) and selection techniques [26].

In parallel to ES and EP, another EA called Genetic Algorithms was being developed and has since become the most popular type of EA in use today.

3.3 Genetic Algorithms

Genetic Algorithms (GA) gained its popularity through the work of John Henry Holland in the 1970s [91]. Continuous theoretical research eventually led to the release of the first commercial GA products in the mid-1980s and development continues to this day.

GAs use a linear binary genetic representation, most usually an array of bits. Each of these binary strings, referred to as chromosomes or genotypes, encodes a solution (phenotype) to an optimisation problem. The evolutionary search process, outlined in Section 3.2.1, operates on the genotype which is decoded into the phenotypic solution to measure its fitness value, pre-selection.

The GA chromosome is usually of fixed-length. It is subdivided into genes with each gene comprising a fixed number of bits. Each of these

genes can be used to encode a value, e.g. integers or reals, and each of these values could be a numerical parameter to a particular problem.

The cycle of the GA follows the pseudocode shown in Figure 3.1 and is described in more detail below:

1. Generate an initial, random population of N strings
2. Decode each string into its phenotype and evaluate its fitness
3. Select two parents from the population
4. Create two children using the crossover and mutation operators
5. Repeat steps 3 and 4 until N children have been created
6. Replace the existing population with the newly created children
7. Repeat steps 2 to 6 until the termination condition is met

The above sequence of actions form the general skeleton of a GA [127]. When implementing one however, some important choices need to be made.

The highest level problem-specific issues are that of the phenotypic solution's structure and the fitness function. The genotype encodes a range of values which are evolved and then evaluated by a fitness function. The values encoded in the genotype must describe a solution to the problem. The fitness function must be able to return a score based on a solution's ability to solve that same problem. The structure of the phenotype and fitness function are always very problem-specific. Other implementation choices however, can be made based on the general nature of a problem.

Take a lower level detail such as the genetic representation for example. The standard GA representation is a binary string, but one may choose to use reflected binary code (Gray code) to avoid the Hamming

Cliff problem that can occur when bit-strings are used to represent integers [25]. Variable-length representations are also possible but require more complicated crossover operations (see Section 3.3.2). The representation's type is also changeable. The standard GA representation is an array of bits, but arrays of other types may also be used, e.g. floating point. These choices are low-level and beyond scope. Other choices in a GA implementation however, are common to most popular EAs.

As can be seen in the list of steps involved in a GA, individuals must be selected, recombined (crossover and mutation) and replaced in the population. The variation operators involved in the EA recombination process are described in upcoming Sections 3.3.2 and 3.3.3 in terms of GA implementation, and later in relation to Genetic Programming and Grammatical Evolution. The selection and replacement techniques described next however, are common to each of these EAs.

3.3.1 Selection methods

At each generation in the EA process, a proportion of individuals from the current population are selected from which to create a new generation. These selected individuals may be bred together and their offspring entered into the next generation. Certain selected individuals may be mutated before being placed back into the population. On some occasions, individuals that have the highest fitness in a population are copied directly into the next generation in so-called elitist selection.

The common factor that determines whether an individual is selected or not is fitness; a fitter solution is more likely to be selected. Several different types of selection method are available. The majority of these

methods are stochastic and purposefully allow for the possibility that a few of the less-fit solutions in a population may get through to the next generation. This randomness promotes diversity in the population and can prevent premature convergence upon inadequate solutions. Roulette wheel selection and tournament selection are the most popular selection methods and are briefly described here.

Prior to the first step in roulette wheel selection (fitness proportionate selection), a fitness value is assigned to each individual. In an analogy to a roulette wheel in a casino, each individual can be thought of as having a portion of the wheel assigned to it based on its fitness. Each individual's fitness value is normalised by dividing its actual fitness value by the sum of all the other individual's fitness values. The sum of all the individual's normalised fitness values is 1. Analogous to a spin of a roulette wheel, a random number is chosen between zero and one. The individual whose normalised fitness is the first value greater than the random number is chosen. In roulette wheel selection the probability of being selected is given by Equation 3.1.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{3.1}$$

where f_i is the fitness of individual i in the population, p_i is the probability of that individual being selected and N is the number of individuals in the population.

Roulette wheel selection may cause some of the higher scoring individuals to be removed from the population. It may also select some of the

weaker individuals, however, this is advantageous as it promotes diversity. Another popular selection method which can exhibit less stochastic noise is tournament selection.

In tournament selection, a number of individuals are randomly picked from the population. "Tournaments" are held on these subsets of individuals to select an individual for crossover. An adjustable tournament size controls selection pressure with larger tournament sizes favouring stronger solutions. The general steps of the tournament selection process are listed below:

1. Choose N individuals from the population (N is tournament size)
2. Choose highest fitness individual with probability p (where $p = 1$)
3. Choose next highest fitness individual with probability $p * (1 - p)$
4. Choose next highest fitness individual with probability $p*((1-p)^2)$
5. Choose next highest fitness individual with probability $p*((1-p)^3)$
6. Continue as required

Tournament selection is very popular in practice. It is easily implemented, the selection pressure is constant and is simply adjusted. In roulette wheel selection, evolution rate depends on a population's fitness variance. This implies that while selection pressure may be high early on in the process, as fitness values become more homologous in later generations, selection pressure drops off possibly leading to premature convergence.

Both roulette wheel and tournament selection are simple, popular selection techniques amongst a large variety of documented and tested methods. As with many choices in an EA setup, the choice of selec-

tion technique can depend on the problem. Difficulties with convergence rate and local optima can be improved through different combinations of selection, variation and replacement operators.

Replacement operators will be discussed in Section 3.3.4. Before this however, two variation operators, namely crossover and mutation are described.

3.3.2 Crossover

The concept of chromosomal crossover is discussed in Appendix Section A.1.2. This process produces genetic variation in a population through the exchange of genetic material between homologous chromosomes and is the inspiration behind the fundamental GA crossover operator.

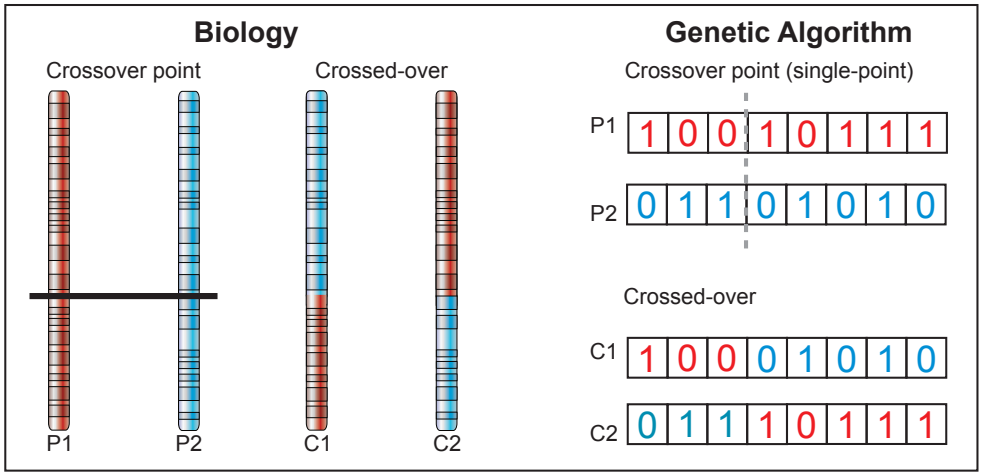


Figure 3.3: The images to the left depict crossover between two biological chromosomes (P=parent, C=child). The right of the diagram shows a single-point crossover between two GA chromosomes.

The relationship between biological and GA crossover is illustrated in Figure 3.3. The images to the left of the figure illustrate the crossover process between two parent chromosomes. The resulting child chromo-

somes each contain a different combination of genetic material from each parent. The GA equivalent is shown to the right of the figure, where each GA chromosome is a binary string. A crossover point is chosen at random and all data that occurs in the string beyond that point is swapped between the two parent chromosomes, producing two child chromosomes. This is an example of single-point crossover.

A variation of single-point crossover called cut and splice uses separate crossover points for each parent chromosome. The bits that occur after these crossover points are swapped, as in single-point crossover, usually producing offspring individuals of differing lengths.

Multiple crossover points can also be applied to each parent. In two-point crossover for example, two points along both parent chromosomes are randomly chosen. The bits that occur between these two points are then swapped between parents to produce the offspring.

Another alternative crossover technique is called uniform crossover where each offspring receives equal amounts of genetic material from each parent. Each bit of the two parent chromosomes are compared and then swapped with a probability of 0.5.

All of these techniques introduce variety into the population, without which the evolutionary engine cannot function correctly. Another great source of variety in both biological systems and GA is mutation.

3.3.3 Mutation

As with crossover, mutation operators are based on the biological mutation process (described in Appendix Section A.1.2). In the biological process, mutation causes a change in the order of bases in a DNA se-

quence. Analogous to this, the GA mutation operator changes the state of a bit in a GA chromosome with some probability, thus introducing genetic diversity into a population.

For a GA, the most simple form of mutation is called bit-flip. The mutation algorithm steps along the chromosome, flipping each bit with some probability. Although in biology there are many types of mutation (Figure A.3 of Appendix Section A.1.2), the bit-flip mutation most resembles a point mutation. In biological point mutation, a base nucleotide is replaced with another.

Figure 3.4 illustrates bit-flip mutation. In this example, a string of 32 bits are mutated with a probability of 0.1 (a much higher probabililty than would be used in practice). The mutated bits are shown in red.

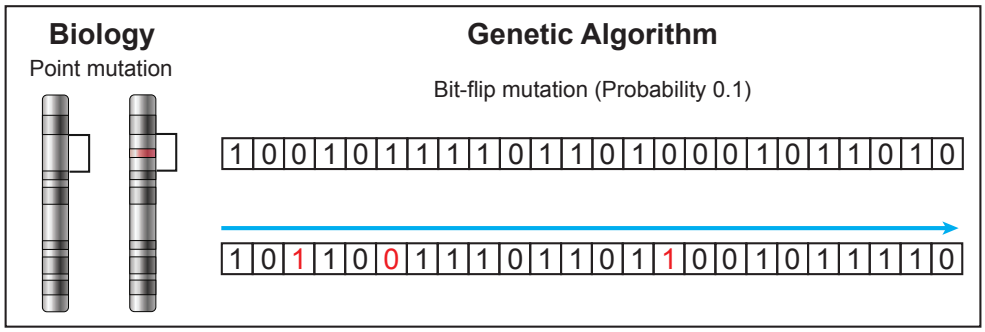


Figure 3.4: An illustration of a biological point mutation is shown to the left of the figure. An example of GA bit-flip mutation is shown to the right. The 32-bit chromosome is stepped through linearly, flipping bits (shown in red) with probability 0.1.

The purpose of mutation in GA is to introduce variety, however, the amount of novel solutions introduced to a population must be restricted to prevent the search becoming random. An appropriate amount of mutation should prevent the search from becoming stuck at local minima and prevent severe slowdown of evolution due to lack of diversity.

The issue of crossover and mutation operators will be explored further in subsequent sections in relation to Genetic Programming and Grammatical Evolution. Before this however, an overview of the replacement strategies used in most major EAs concludes this section on GA.

3.3.4 Replacement strategies

When new individuals are created, they become part of the next generation. These new individuals must replace individuals of the previous generation and there are several strategies for this replacement: generational, steady-state and elitism.

In generational replacement strategy, parents are replaced by their children regardless of fitness. An alternative to this is steady-state replacement in which every two parents produce two children. One of the parent individuals is only replaced by its child if that child has a better fitness score than the worst scoring parent.

An elitist strategy may also be used in tandem with either of these two techniques. In this strategy, individuals with the best fitness in a generation are kept on as part of the next generation. All of the remaining individuals are subject to whatever other replacement strategy is in effect (generational or steady-state).

GA conclusion

The popularity and success of GA has spawned large amounts of research, refinements, specialisations and related techniques. These other techniques have enjoyed varying amounts of success. One particular technique called Genetic Programming which focusses on the evolution of

computer programs rather than function parameters is discussed in the following major section.

3.4 Genetic Programming

Like GA, Genetic Programming (GP) follows the EA paradigm. While GA evolves the numerical parameters of some problem, GP evolves computer programs to perform a user-defined task [22].

The origins of GP can be traced back to Nils Aall Barricelli's work with evolutionary algorithms in the 1950s [23]. In the 1960s, Lawrence J. Fogel used evolutionary algorithms to discover finite-state automata [60]. Evolutionary algorithms continued to develop throughout the 1970s through the work of Ingo Rechenberg amongst others [167]. The modern tree-based method of GP was first described by Nichael L. Cramer in 1985 [43].

In the 1990s, John R. Koza pioneered the use of GP for the optimisation of complex problems [105]. The power and applications of GP have grown significantly in the 2000s as available computational power grew exponentially. GP is now considered a cutting edge search technique that has been used in electronics design, hardware and software design and game playing, amongst many others [144, 111, 106, 121].

The computer programs that GP evolves are typically represented as tree structures. Mathematical expressions encoded in these trees are easy to evolve and recursively evaluate, as every node in the tree structure is an operator function and each of the terminal nodes are operands. Traditionally, languages such as LISP which naturally embody tree structures

have been used as the GP representation [158]. Certain GP implementations have also used non-tree representations successfully.

Figure 3.5 presents two example functions. The functional representation is shown above each tree and its corresponding function shown beneath. The tree on the left represents a simple mathematical expression while the tree on the right incorporates a conditional statement. Note that the *if* function has three arguments (operands), referred to as its arity.

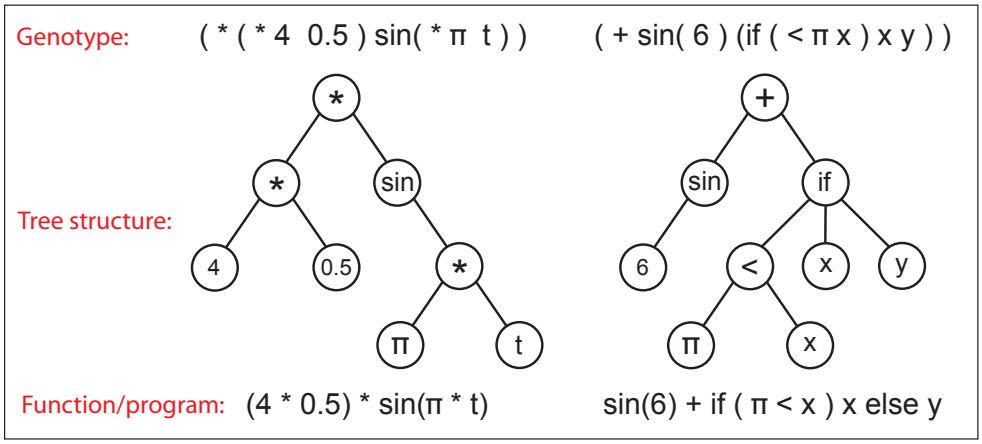


Figure 3.5: Tree representation of two functions. The tree on the left represents a mathematical function. The tree on the right includes a conditional operator.

A full introduction to genetic operators and GP in general can be found in the literature [22, 158]. In this thesis, a simple overview of the standard GP operators is provided. The selection schemes and replacement strategies described in Sections 3.3.1 and 3.3.4 equally apply to GP, with tournament selection being the most common.

What does differ from GA is the recombination and mutation processes used in GP, as they must deal with the added complexity that tree structures introduce. The crossover and mutation techniques used

in GP are described in an upcoming section. Before this however, two GP initialisation methods are discussed.

3.4.1 Initialisation

GP initialisation is more complicated than that of GA. In GA, initialisation can be simply a case of randomly creating a population of binary strings. Whilst the individuals in the initial GP population are also usually random, as tree-based representations are used, the situation is more complex.

Each individual is a syntax tree and the size of each tree is variable, i.e. both the structure and content of the tree must be specified and evolved. When a tree is created, the maximum depth of the tree determines the maximum size of the program and a maximum tree depth parameter must be set prior to initialisation. The two simplest types of initialisation are called full and grow.

Assuming the root node of a tree is at depth 0, the depth of each node in a tree is calculated as the number of edges between it and the root node. The depth of the entire tree is determined by the leaf node with the greatest depth.

Initial populations created using the full initialisation method consist entirely of full trees; that is all the leaves are at the same depth. An example of a tree created using the full initialisation method is shown in Figure 3.6. Each node is created by randomly selecting a function from the function set until the maximum tree depth is reached. The leaves of the tree are then randomly chosen from the terminal set.

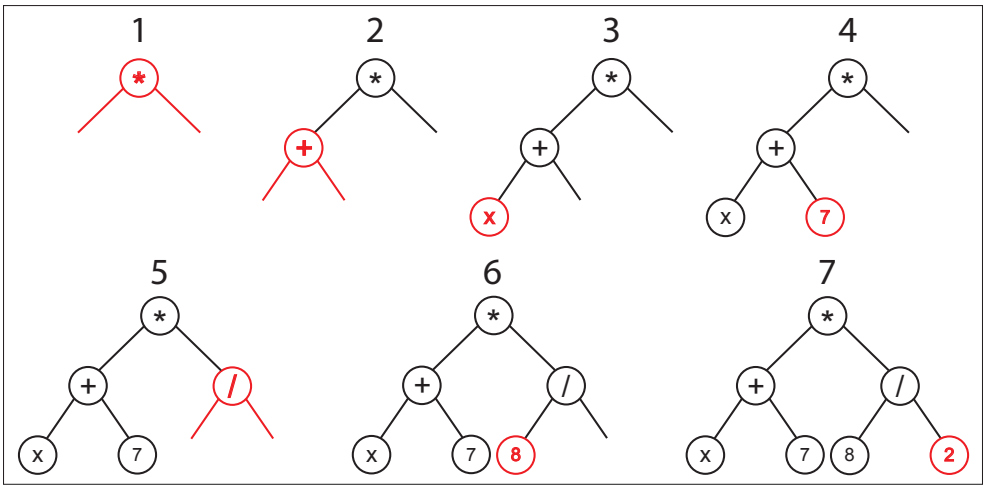


Figure 3.6: Full initialisation: steps taken in creating a tree with a maximum depth of 2.

In a function set in which all functions have the same arity, all trees created using full initialisation will have the same number of nodes and shape. Primitive sets including mixed-arity functions introduce variance in the number of nodes and shape of a tree. This variety is limited however, and motivates the use of the grow initialisation method.

In grow initialisation, each node in the tree is selected from the entire primitive set including both functions and terminals, until the maximum depth is reached. As with full initialisation, at the depth limit only terminals may be selected. Figure 3.7 shows the creation of a tree using grow initialisation. By allowing terminal nodes to occur at non-leaf nodes, entire branches of the tree can be closed without reaching the maximum depth. Because of this, the size and shape of trees can vary greatly, leading to an initial population of highly varying programs.

To ensure that a wide variety of initial population individuals are created, that span the range of trees generated by both full and grow initialisation methods, Koza proposed ramped half-and-half initialisation

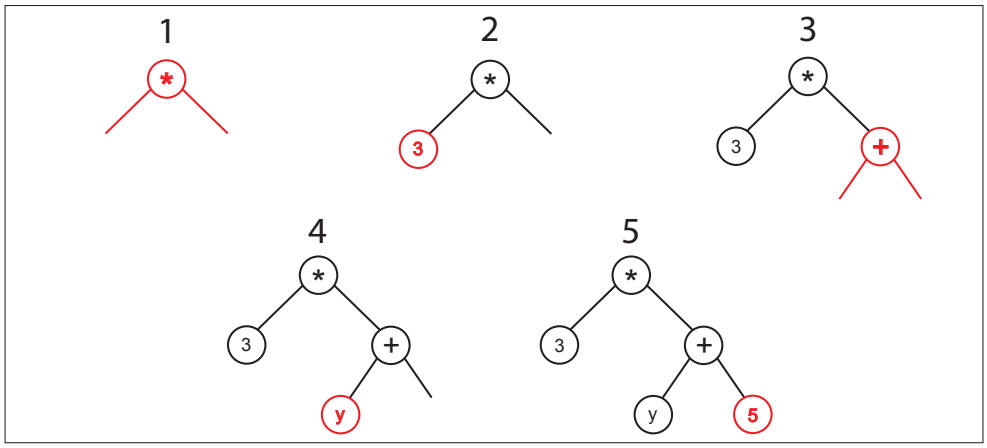


Figure 3.7: Grow initialisation: steps taken in creating a tree with a maximum depth of 2.

[105]. This method involves creating half the population using full initialisation and the other half using grow initialisation. The term ramped refers to the use of a range of depth limits, further varying the shapes and sizes of the initial population.

These initialisation methods are popular due to their ease of implementation and use. The behaviour of grow and full initialisation is highly dependent on the number of terminals versus functions available and also the arity of those functions.

Other initialisation techniques are available which address specific issues in GP and it is possible to create initial populations whose trees embody some domain knowledge rather than being completely random. The most important aspect of initialisation is variety generation. Without variety, the evolutionary process would cease to function. In the following section we discuss one of two operators which generates variety throughout the evolutionary cycle.

3.4.2 Crossover

Crossover in GP involves the switching of subtrees between individuals in the population. As GP uses a tree-based representation, when a node is replaced, an entire branch of the tree may be replaced. The replacement of an entire branch can create a big difference between the expression encoded in the pre-crossover chromosome and that of the post-crossover chromosome. This is called subtree crossover and it is the most common type of crossover used in GP.

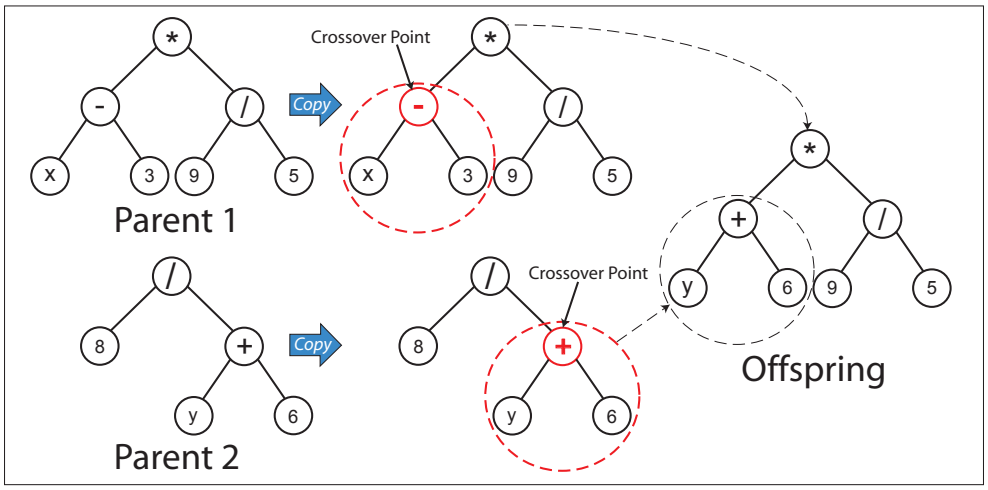


Figure 3.8: Illustration of GP subtree crossover.

An example of subtree crossover is shown in Figure 3.8. The diagram illustrates the 4 steps of the GP subtree crossover process, as listed below:

1. Select 2 parent individuals from the population
2. Randomly pick a node from Parent 1
3. Randomly pick a node from Parent 2 (independent of Parent 1)
4. Create a new individual from the subtrees at the crossover points

As can be seen in Figure 3.8, the offspring individual is created as the subtree rooted at the crossover point in a copy of Parent 1 is replaced by a copy of the subtree rooted at the crossover point in Parent 2. The use of copies of the original parents allow them to participate in multiple crossover processes, retaining their original state.

Crossover points are not usually selected with uniform probability. Typically the majority of nodes in GP individuals are leaf nodes and as such, randomly selecting crossover points from the entire tree can lead to a large number of crossover operations that swap very small subtrees or leaves. It was suggested by Koza that when picking crossover points, functions should be chosen 90% of the time, and leaves only 10% of the time [105]. This approach to crossover point selection has become popular.

As with the initialisation techniques discussed in Section 3.4.1, this form of subtree crossover is one of a large variety of possible approaches. Similarly, the following section describes two popular mutation techniques, mutation being the other variation operation used in GP.

3.4.3 Mutation

In GP mutation, the detail of a particular node may be modified or the entire node replaced, affecting the tree structure.

The most common form of mutation in GP is subtree mutation. In this form of mutation, as shown in Figure 3.9, a subtree rooted at some randomly chosen point in the individual is replaced by a newly generated subtree. The steps of the subtree mutation process are given below:

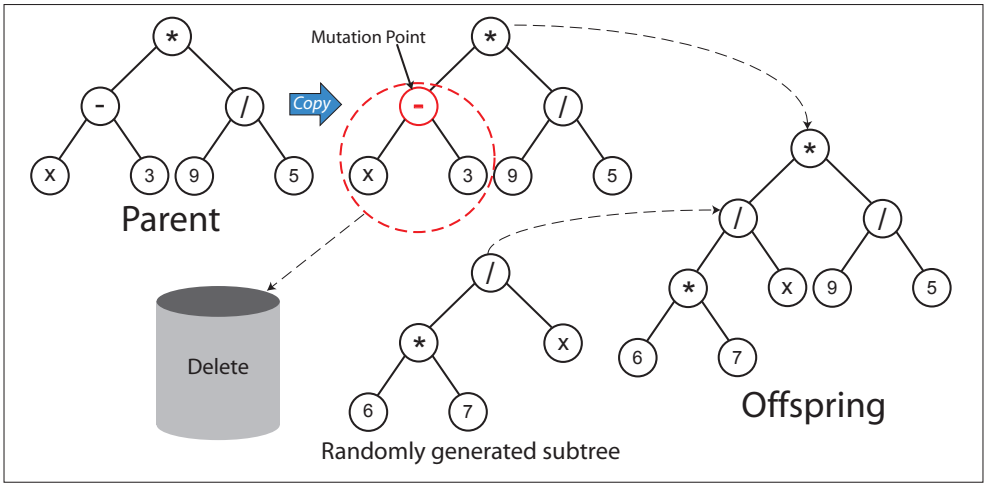


Figure 3.9: Illustration of GP subtree mutation.

1. Select 1 parent individual from the population
2. Randomly pick a node from that parent
3. Remove the subtree rooted at the selected mutation point
4. Grow a new subtree according to an initialisation technique
5. Place new subtree at the location of the previously removed subtree

Another commonly used technique is point mutation in which the primitive stored at a randomly selected node is replaced by another randomly chosen primitive. The nature of the original primitive must be taken into account and the replacing primitive must be of the same arity. If the nature of the node on which mutation is to take place is not taken into account, all operations must be made fail-safe.

During point mutation, each node in the individual is visited and mutated with some probability allowing multiple nodes to be changed in a single mutation process. When an individual is being subjected to subtree mutation however, only a single subtree is replaced.

The actual rate at which operators such as mutation and crossover are applied is variable. Mutation operators are often applied to less than 1% of the individuals in a population per generation. In contrast, crossover rates are relatively high; often set at 90% or more. Depending on mutation and crossover rates, another operator called reproduction simply takes individuals of high fitness and copies them into the next generation.

GP conclusion

The choice of operators and operator rates can be problem-specific and depend on the GP implementation. The system described in the preceding sections is the most basic GP implementation which was developed in the late 1980s and early 1990s. Since then many alternative GP implementations and extensions have been proposed [158] and a number of open issues remain [151]. In the next section, one particular specialisation of GP called Grammatical Evolution is explored.

3.5 Grammatical Evolution

Grammatical Evolution (GE) is a relatively new type of EA [150, 47]; a grammar-based specialisation of GP [121].

In GE, the genetic representation is a variable-length binary or integer string. The evolutionary operators are applied to this genotype, rather than the phenotype, and this separates the search and solution space. More recently, operators are also applied to the derivation trees [80].

It is this separation that distinguishes GE from most GP systems and make it attractive for representing solutions to highly complex problems. The efficiency of the evolutionary search may also improve over an equivalent GP search, as the genotypes on which the GE search is performed are abstractions of phenotypic solutions into a more compact form.

The use of a grammar facilitates this search-solution space separation. It also allows for the incorporation of problem domain knowledge and constrains the search. This is an important quality as GE's performance is dependent on its search space size and it is desirable to minimise it.

The grammar's actual role in the GE process is to map genotypic strings to a function or program. This process is described in a following subsection. Before this however, the implementation and modular approach of GE is discussed.

3.5.1 Implementation details

GE is modular in its implementation, as shown in Figure 3.10.

A GE system requires a grammar, a fitness function and a search engine. The fitness function can take any form as long as it can accept a phenotypic solution and return a numerical score back to the GE system; a lower score implying a better solution. GE's modularity provides great flexibility as the fitness function can be of any level of complexity. The fitness function can even be a large scale simulation application, as will be discussed in relation to a real-life application in Section 8.1.2.

The search engine used in the system is also modular. As the GE search takes place on the same objects as those used in GA, the search aspect of GE may be left to any existing variable-length EA search imple-

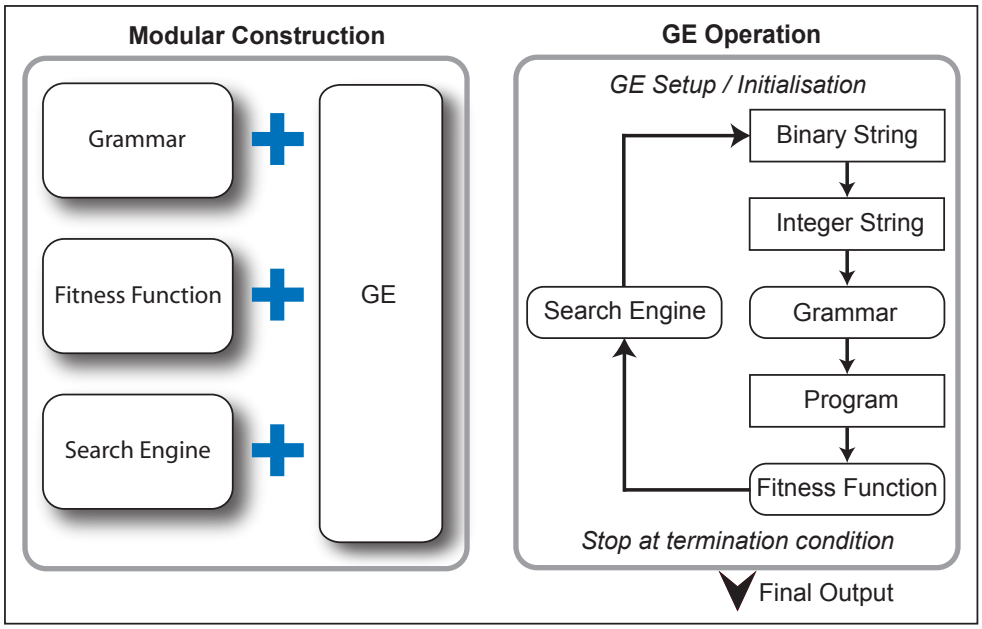


Figure 3.10: The modular construction of a GE system (left) and the operation of a GE system (right).

mentation. Other search techniques may also be employed for example, Particle Swarm Optimisation has been successfully used leading to a hybrid system known as Grammatical Swarm [140] and Differential Evolution (DE) has been used to create a Grammatical Differential Evolution (GDE) system [139].

On the right side of Figure 3.10, the GE operation cycle is shown. The system starts with the system setup and population initialisation. As the initial population is a set of binary, or more recently integer strings, initialisation is performed, as described in upcoming Section 3.5.4. For each string in the population, through a series of transcriptions and translations using a grammar (Section 3.5.3), a phenotypic function or program is created. These phenotypes are passed to the fitness function module which returns a fitness score.

Once all of the individuals in a population are scored, the search engine module performs its task. The general EA process outlined in Figure 3.1 of Section 3.2 is followed according to some specific EA search implementation. Individuals are selected based on the fitness scores, recombined and replaced in the population ready for the next generation.

From Figure 3.10 it can be inferred that a person attempting to create a GE system need only implement the mapping process from the integer list to the program tree. The mapping process will be discussed in detail in Section 3.5.3. Before this however, it is necessary to explore the form of the grammar used in GE.

3.5.2 Grammar

The GE grammar is context free and is typically expressed in Backus Naur Form (BNF). BNF is a notation technique for exactly describing the syntax of a computer language.

In GE, the BNF representation is used to state a set of production rules which make up a complete grammar. The BNF grammar is composed of the tuple <T, N, P, S> as presented below (example values are shown in the curly brackets).

- T is Terminals set { *Sin*, *Cos*, *Tan*, *Log*, $+$, $-$, $\div$, $\times$, *X*, (,) }
- N is Non-terminals set { *expr*, *op*, *pre-op* }
- S is Start symbol (e.g. S = <expr>, also a member of N)
- P is Production rules set (described below)

A production rule is in the form <symbol> ::= *some_expression* where <symbol> is a non-terminal such as those above and *some_expression* is a

sequence of one or more symbols which can be non-terminals or terminals.

The right hand side of the production rule may contain more than one expression, each separated by a vertical bar, '|', indicating a choice. A chosen expression is essentially substituted for the symbol shown on the left hand side of the production rule.

```
<expr>  ::=  <expr> <op> <expr>        (A)
           |  ( <expr> <op> <expr> )   (B)
           |  <pre-op> ( <expr> )      (C)
           |  <var>                    (D)

<op>  ::=  +  (A)
         |  -  (B)
         |  /  (C)
         |  *  (D)

<pre-op>  ::=  Sin  (A)
             |  Cos  (B)
             |  Tan  (C)

<var>  ::=  X  (A)
```

Listing 3.1: Basic grammar with common examples of production rules.

The basic grammar shown in Listing 3.1 includes common examples of production rules. The parenthesised letters at the right hand end of each production rule expression are not part of the BNF grammar. They indicate that each expression represents a choice for the corresponding terminal or non-terminal.

These production rule choices are fundamental to GE's aforementioned translation process. The values in a genotypic integer string are used in sequence to choose rules from one of these BNF grammars, thus constructing the phenotype. This mapping process is discussed in the following section.

3.5.3 Genotype-phenotype mapping

As previously stated, GE performs its evolutionary operations on the genotype. This genotypic string representation describes a solution as DNA describes a protein. A grammar is used to expand this genotype into a phenotype, similar to how DNA is expanded into a protein. This is called the genotype-phenotype mapping process and is illustrated in Figure 3.11.

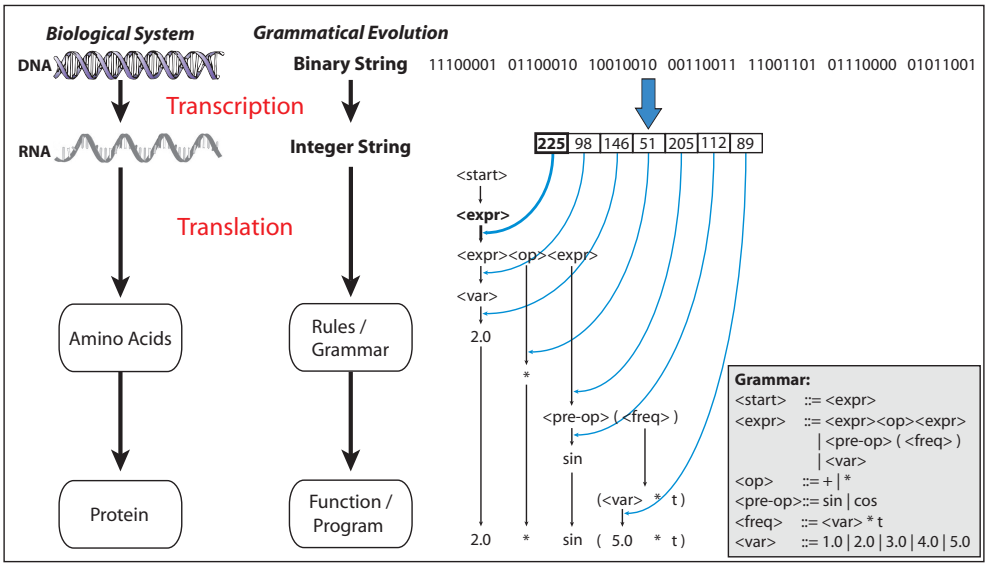


Figure 3.11: Genotype to phenotype mapping process with GE example.

In this figure it can be seen that there are correlations between the steps in a biological system's genotype to phenotype mapping, and that of GE. The binary string representation is analogous to a DNA sequence and the conversion of the binary string to an integer string is comparable to the transcription process described in Appendix Section A.1.1. The translation process that occurs in a biological system, in which mRNA is converted to an amino acid chain that eventually becomes a protein, is also mimicked in GE.

In genetics, a codon is a group of three nucleotides in a DNA sequence (chromosome) which describes an amino acid (Appendix Section A.1.1). In a GE chomosome, each binary or integer string contains many codons and each of these codons map to a single production rule choice in the grammar.

A single binary string codon for example, might be 8-bits long and therefore transcribes to integers between 0 and 255. A GE implementation whose genetic representation is an integer string may have a much larger range of values, e.g. Java integer, 2^{32} bits. For simplicity, the explanatory examples in this thesis assume an integer range of 2^8 bits.

The process of converting the integer string into a function or program via the grammar, is equivalent to the biological translation process. The length of the GE chromosome itself is variable and usually capped at some maximum length. Depending on the implementation, when the end of the chromosome is reached during the mapping process, the operation can wrap around back to the start of the chromosome and continue translating.

An example of the conversion process from binary string to phenotypic function is shown to the right of Figure 3.11. Firstly the binary string is transcribed into its integer equivalent. Then starting at the left of the integer chromosome, the integer values are translated into elements of a function or program using the displayed grammar. The grammar in the figure contains terminals and non-terminal sets that are sufficient to produce summations of sine and cosine waves that are functions of a free variable called t and have a fixed range of amplitudes and frequencies.

The translation process shown in Figure 3.11, begins at the start symbol which has a single rule, the <expr> non-terminal. This non-terminal contains three production rules. To select one of these rules, the leftmost integer value is used, as highlighted in the figure. The integer value is 225 and there are 3 possible rules. The following calculation is performed: 225 mod 3 = 0. This result indicates that rule 0 should be chosen, which is <expr><op><expr>. This process of choosing production rules based on the integer string continues as indicated in the figure until the derivation is complete and the encoded function or program is obtained.

It should be noted that each codon is polymorphic; its interpretation depends on its context. In the above example, when mapping the <expr> non-terminal, the calculation performed is 225 mod 3 = 0. If the same integer is used to map the <pre-op> non-terminal, the calculation would be 225 mod 2 = 1.

For a deeper understanding of this translation process, the translation of the integer string can be visualised using a derivation tree, shown in Figure 3.12. In this figure an integer string is translated into its phenotypic function. The derivation process is shown on the left of the figure consistently following the same process: select the current integer i in the string; find the number of production rules n of the current non-terminal; calculate i mod n; the calculated value determines which of the current terminal or non-terminal's rules is to be selected.

As the derivation process progresses, the derivation tree on the right of the figure grows. The figure also demonstrates the wrapping process. If the end of the integer string is reached before the derivation process has

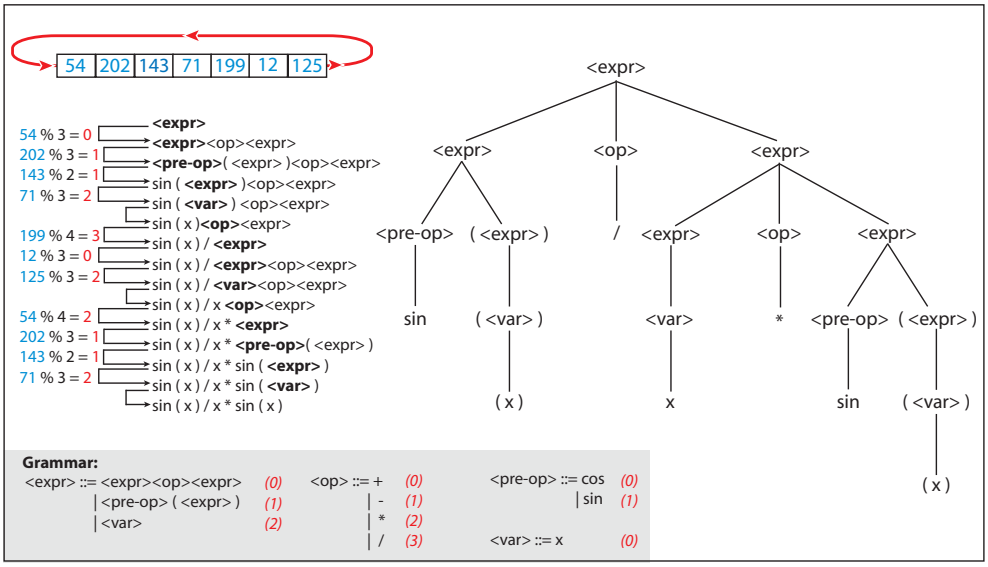


Figure 3.12: The derivation tree illustrates the translation process. The integer string is only 7 codons in length and therefore wrapping is employed to produce a fully derived tree.

completed, i.e. a fully formed tree has not been derived as non-terminals remain at the leaves, the algorithm wraps around and continues to step through the integer string from the beginning. In GE there is usually a limit set to the number of times the algorithm may wrap around. If this limit is reached without producing a fully derived tree, that individual is declared invalid. Once a function tree is fully derived however, it can be visualised as a tree structure similar to the GP tree representation displayed in Figure 3.5 of Section 3.4.

The genotype-phenotype mapping process is the main distinguishing feature of GE. Other mapping variations are also possible, such as in πGrammatical Evolution's position-independent variation on the GE mapping process [141, 55, 63].

Regardless of the mapping process employed, the selection and replacement techniques can be identical to those of GA (described in Sec-

tion 3.3.1 and 3.3.4 respectively). Other GE processes such as the variation operators, while perhaps handled by the GA search module, actually behave as a hybrid of the GA and GP operators. This will be discussed in a subsequent section.

The initialisation techniques for GE also follow this hybrid approach and are briefly described in the following section.

3.5.4 Initialisation

A simple choice of initialisation technique for GE would be randomly generated integer strings. This approach has the drawback that, with certain grammars, individuals could either be very short or have no guarantee that they will terminate (invalid individuals).

Ideally the initial population would contain individuals of a range of sizes. In Section 3.4.1, the GP ramped half-and-half initialisation method was described and the most popular initialisation technique in GE is based on this.

Both a start depth parameter and a maximum depth parameter must first be defined. To ramp the depths of trees in the initial population, a third variable called the current maximum initialisation depth is calculated as a value that increases incrementally from the minimum depth to the maximum depth parameter.

Half of the individuals are created using the full tree generation method in which only rules that lead to the creation of trees of the maximum depth are chosen. The other half are created using the grow method in which rules are randomly chosen until the current maximum initialisation depth is reached.

Similar to initialisation, the GE crossover technique may at first glance appear the same as the previously described GP subtree crossover, however, an analysis of the derivation process shows otherwise.

3.5.5 Crossover

Subtree crossover in GP was discussed in Section 3.4.2. The process in GE is similar in some ways, but as the crossover operator is applied to the genotype, the resultant children are not simply copies of the parents with a particular subtree swapped [143].

In Figure 3.13, the derivation trees of two parent individuals are displayed. The integer string and grammar are also shown. The parent individuals are subjected to single-point crossover, as indicated by the dual colours present in the offspring's chromosomes. This example of genotypic crossover is the cut and splice GA method described in Section 3.3.2. A single point is randomly selected on each of the parent chromosomes. As can be seen in the figure, although both parent chromosomes are of the same length, the randomly chosen crossover points indicated by a dashed line produce children of differing length. Due to wrapping, the lengths of the chromosomes do not influence the size of the derivation trees, as can be seen in this example.

By stepping through the derivation trees of the children, it is clear that applying the crossover operator to the genotypes does not provide a straight subtree swap. The phenotypes of each of the individuals present are also displayed in Figure 3.13.

The function trees of these phenotypes are shown in Figure 3.14. This figure shows both the similarities and differences between parent

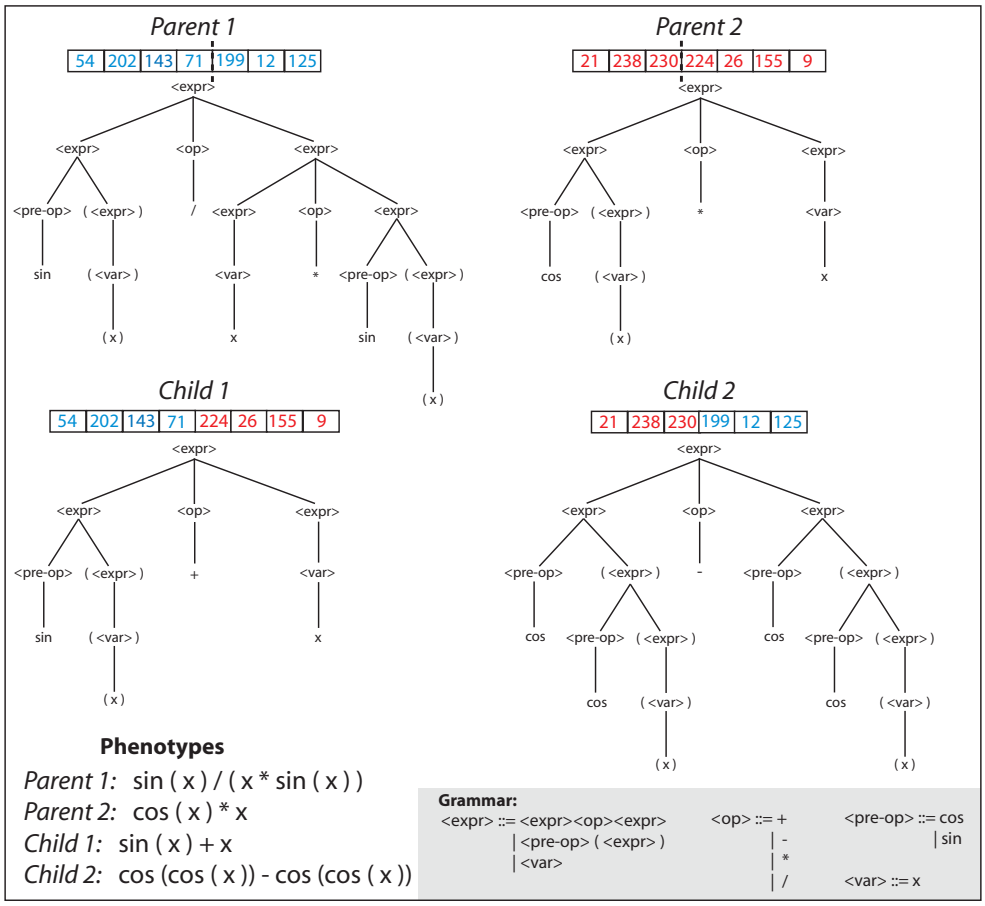


Figure 3.13: Illustration of GE single-point crossover with corresponding derivation trees.

and child individuals. This is due to the variation operator being applied to the genotype, as restated by the inset diagram.

In GE, it is possible to emulate the GP subtree crossover method using a restricted form of two-point crossover [80], however, the GP subtree crossover has been subject to criticism. It has been suggested that GP's use of trees negates its use as a biological search method [42]. It has also been proposed that GP subtree crossover is only marginally more effective than a system which inserts randomly generated subtrees [13].

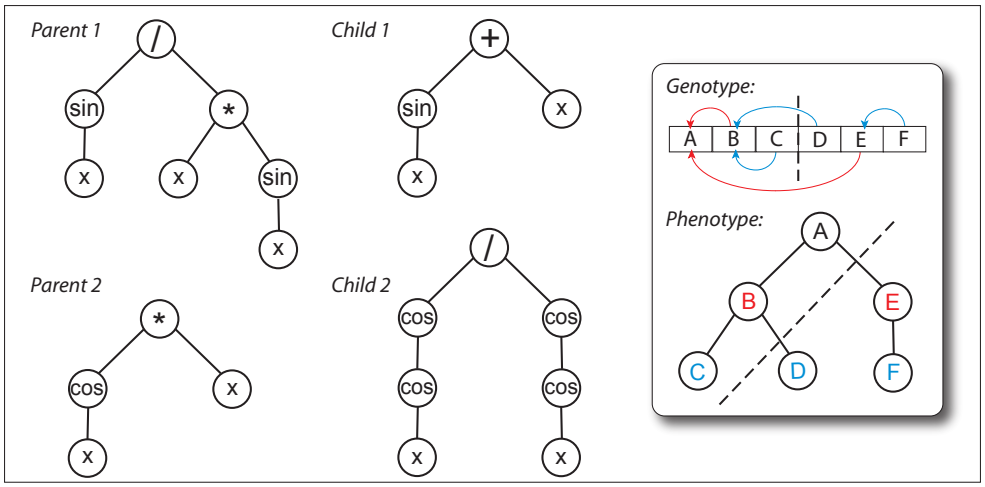


Figure 3.14: Illustration of GE single-point crossover with corresponding function trees. Note that the crossover operation acts on the genotype as illustrated in the inset image to the right of the figure.

As has been pointed out above, the GE and GP crossover differ significantly as GE performs its operations on the genotypic chromosome. The GA inspired single-point crossover employed however, has been labelled a destructive crossover operator and criticised accordingly. O'Neill and Ryan detail extensive experimentation on the single-point crossover method and criticisms of it [150]. They conclude that a biologically inspired homologous crossover operator designed for GE performed no better than the standard single-point crossover. Further to this, they debunked any suggestion that the single-point crossover operator is only as useful as crossing over randomly generated subtrees [150].

For the purposes of this thesis, we accept these experimental findings and the conclusion that, in the GE context, the single-point crossover operator is a very powerful variation operator that drives the GE evolutionary process. The mutation operator has a lesser but still important role to play in the GE evolutionary process and is discussed next.

3.5.6 Mutation

The mutation operator used in GE is int-flip as standard. This is identical to the GA bit-flip mutation method described in Section 3.3.3 and illustrated in Figure 3.4, except that int-flip operates on an integer string representation; each codon (integer) is replaced with uniform probability by a new integer within the set integer range.

The effect of an int-flip mutation on the derivation tree is shown in Figure 3.15. It can be seen that changing a single integer in the original chromosome can change the phenotype significantly [85].

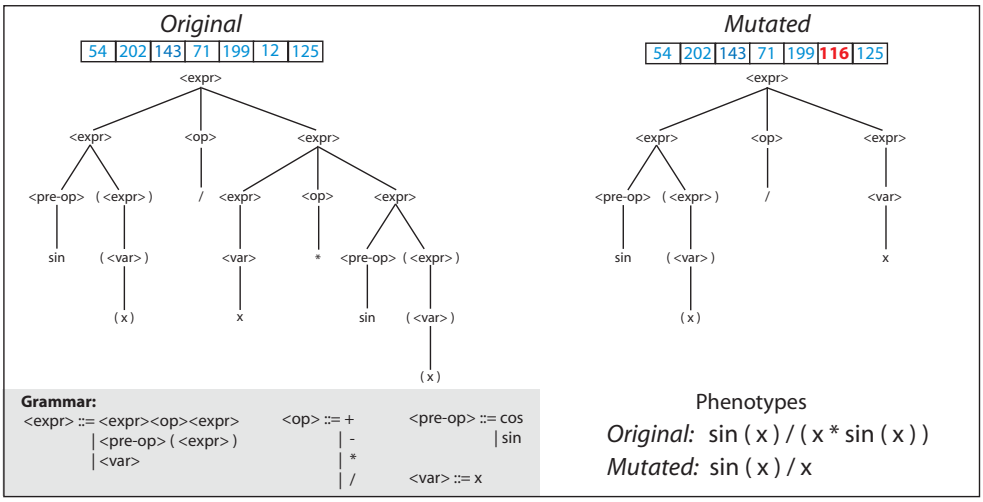


Figure 3.15: Illustration of GE int-flip mutation with corresponding derivation trees. A mutation is highlighted in red.

In Figure 3.16, the phenotypic function trees from the example in Figure 3.15 are shown. The effect of mutating a codon in the chromosome ripples down through the following codons, as the derivation of a codon depends on those codons that preceded it. In this example, changing a single integer in the chromosome terminates the tree's growth and an entire subtree from the original individual is replaced by a single terminal.

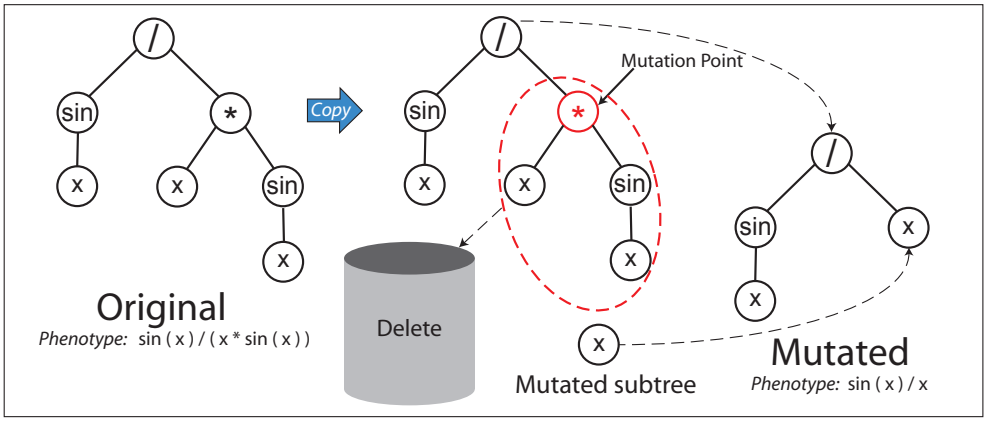


Figure 3.16: Illustration of GE int-flip mutation with corresponding function tree.

This example of mutation is a good illustration of how a small change to the GE genotype can lead to a large change in the phenotype. This feature of GE can have positive and negative results on the evolution process, depending on the problem domain.

GE conclusion

The description of GE presented here is only an introduction to a topic on which research is ongoing. The exploration into different mapping processes such as πGE, mentioned previously [55], continues and a variety of different grammar types including tree adjunct grammars are explored in the literature [142, 131, 153, 152, 37]. Topics such as the genotype-phenotype mapping process [56], grammars [184] and mutation [34] are analysed whilst alternative crossover techniques and their use with GE are often proposed [120].

In the preceding sections, we have described the modular implementation of GE, grammars, the genotype-phenotype mapping, initialisation and the variation operators employed in a standard GE implementation.

Where applicable, these topics have been discussed with reference to both GA and GP, in terms of similarities and differences. What follows is a discussion on the strengths, weaknesses and applications of the presented EAs, concluding the natural computing chapter.

3.6 Chapter summary

GA is currently the most popular EA in use today. It has been applied to a large variety of problem domains including social systems, population genetics, economics and various optimisation tasks [127]. It has also been applied to animal locomotion pattern generation as was discussed in Chapter 2. GA generates solutions by evolving the numerical parameters to some problem. In some situations, this is ideal. In others, a more intuitive phenotypic solution representation may be beneficial.

The objective of both GP and GE is to automatically evolve entire programs to solve some problem. This is a very attractive proposition for many real-world applications for which GAs are not entirely suitable. GP has been effectively used to design hardware, software, electronics and has produced human-competitive results in many other problem areas [106]. Although a relatively new EA, GE has been successfully applied to financial prediction [44, 30], game playing [64], music composition [168] and other applications.

As seen in previous sections, in both Koza/Cramer-style GP and GE, the phenotypic individual solutions are recursively evaluated treelike structures. It is the functioning of the evolutionary search that distinguishes these two systems.

In GP, the genetic operators directly manipulate individuals commonly represented as LISP-style tree expressions. As discussed previously, in GE the genetic operators are applied to an integer string which encodes solutions. These strings are then mapped to their phenotypic function or program through a grammar. This use of a grammar and the accompanying mapping process makes it significantly easier to use GE to search a variety of different structures.

In GE, the separation between search and solution space allows complex problem domains to be encapsulated in a BNF grammar. Through this grammar, the search space can also be restricted and problem domain knowledge incorporated as the composition of the grammar allows one to bias the search and control the structure of the output phenotypes.

The use of a grammar also avoids the problem of the GP function set closure requirement, a drawback of the original type-free GP. This requirement stipulates that each function of the set must be able to accept the output of all other functions in the set as an argument. Whilst more modern GP frameworks solve this problem by using a single data type or type supporting systems with certain limitations, GE avoids this problem entirely through use of the grammar.

Besides GE's power to constrain the search, its main strengths lie in its flexibility and ease of use. Its modularity allows a user to easily experiment with a variety of different search strategies be they evolutionary, deterministic or any other compatible techniques. As mentioned in the previous section, research on many aspects of GE is ongoing and there is a steady flow of new contributions such as mappings, grammar types and especially applications.

This thesis is concerned with exploring the use of EAs for generating animal animations. We have chosen a standard GE implementation, GEVA [148], as the EA to use in this exploration for two major reasons. Firstly, the ability to easily create grammars which constrain the search space and allow problem domain knowledge to be incorporated, make GE an ideal candidate for the variety of experiments undertaken in this research. Secondly, GE's separation of search and solution space render phenotypic complexities inconsequential and the problem domain associated with generating an animal's motion pattern is large and complex. Further application-specific details of the GE implementation, grammars and fitness function will be discussed in relation to specific experiments in Part III.

In the final chapter of Part I, aspects of biology which can be exploited for the production of realistic animal animations are discussed.

Chapter 4

Biology

Animations of a real-life animal can benefit from a knowledge of that animal's anatomy and behaviour [203]. The extent of the biological information required however, depends on the nature of the animation.

Images of animals are so ubiquitous in our everyday lives that audiences are highly familiar with physically realistic animal motions. A morphologically correct model may be visually acceptable in a static scene but when movement is required, a person can easily identify when a motion doesn't "look" or "feel" right [155].

Some situations may simply require an aesthetically recognisable animal. A scene in which an animal is standing still or grazing for example, may merely require a model constructed from external measurements, or at least visual estimates, of a real-life animal's geometry. In contrast to this, if a scene requires a highly detailed animation of an animal in motion, the model must be constrained so that its limbs only move within the physical limits of the real-life animal's joints [204]. In addition to this, its surface geometry can be modified in line with muscle

contractions, breathing and ground impacts [41]. Underlying muscles and connective tissues can also be modelled producing motion involving elastic mechanisms [197].

The research presented in this thesis is concerned with producing animal models that move in a visually and physically plausible manner. Previously mentioned aesthetic issues such as surface deformations are secondary to the utilisation of accurate limb motion patterns during locomotion. To achieve this, we exploit the findings of the biomechanics field and use data measured from real-life animals. Before the issue of motion patterns can be discussed however, some aspects of animal anatomy must be explored.

A basic knowledge of anatomy is important in terms of this research and some fundamental anatomical principles are explored in Section 4.2. Building on this knowledge, details of the equine musculoskeletal system and allometry are presented in Section 4.3. Shifting the focus from structure to motion, in Section 4.4, the gaits (patterns of limb motion) and gait transitions of horses are examined in detail. Following on from this, the biology chapter concludes with an overview of dynamic similarity theory.

In advance of an examination of the anatomy and motion of the modern horse, in the following section, the subjects of equine evolution and selective breeding are briefly discussed.

4.1 Equine evolution and breeding

Modern horses are a product of millions of years of natural evolution and hundreds of years of breeding by humans.

There is huge variation in terms of size and shape between modern breeds of horse, but although a Shetland pony and a Draught horse have very different dimensions, they are both of the same species; the domestic horse (*Equus ferus caballus*).

Realistic animations of horses require realistic models. Horses are bred for differing functions and therefore have varying dimensions, proportions, musculature and temperament. These important differences can be be reflected in the computer model.

The range of differences between horse breeds is a product of artificial selection by humans, which will be discussed in Section 4.1.2. The general function of various breeds of horse shall also be described as this information can be used to create more accurate animations.

In preparation for this discussion of artificial selection, the natural evolution of the modern horse is briefly described.

4.1.1 Natural evolution

Evidence suggests that the modern species of horse evolved from a small dog like mammal that lived over sixty million years ago. Through mutation, variation and natural selection the bones, toes and teeth have changed dramatically over the millennia, yielding the modern horse [117].

Over the course of its evolution, the horse's ancestors moved from a forest setting to expanding grasslands. These grasslands provided ample

grazing for those animals whose teeth could handle tough food. As such, the animal's teeth became bigger and adapted in shape. Through mutual natural selection, plants became tougher, selecting animals with the most durable teeth and the grazing animals encouraged even tougher plants.

As these ancient animals moved completely from the forests to the grasslands, they were in danger from predators and consequently their leg bones began to lengthen to allow for faster locomotion. As the predators themselves became faster, the horse-ancestor's legs further increased in length. Some bones fused together, improving stride power at the expense of rotation. Further improvements in speed came by standing permanently on tiptoe. Unneeded toes and other bones shrunk, became vestigial or disappeared.

As animals of this *Equus* species moved about the globe, environmentally driven adaptations occurred giving rise to zebras, onagers and desert asses. One particular *Equus* species spread across Europe, the Middle East and Asia, which would eventually be domesticated by humans to become the modern horse *Equus ferus caballus*.

The evolution of the modern horse is shown in Figure 4.1. The increase in body size shown in this diagram however, was not a consistently gradual process over time. Based on studies of the fossil record, the evolution contains a long period of relative stasis ($\sim$32 million years) followed a period of change and diversification ($\sim$25 million years) [116].

Whilst the modern horse is a distinct looking animal in many respects, one of its most striking features is its long limbs which lack musculature on the lower limb making them lighter. This adaptation makes the horse excellent at fast running over hard ground.

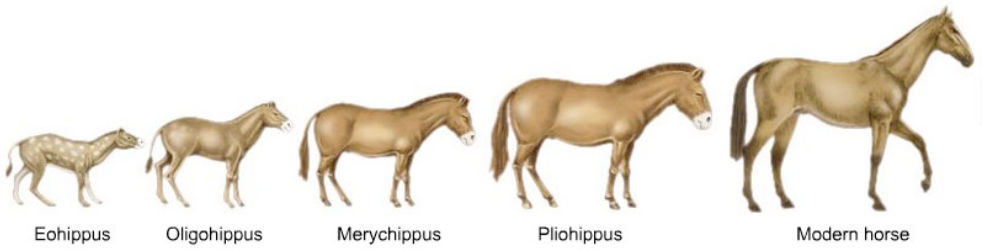


Figure 4.1: The evolution of the modern horse. (Image © National Taiwan Science Education Center [192])

The impressive musculature of the animal's hindquarters gives it great strength. Ever since the horse was domesticated (estimated to be around 4000 B.C.), humans have harnessed the horses' strength to farm the land, and later, its speed for travel and sport. For hundreds of years, humans have been selectively breeding horses for specific activities giving rise to the diverse shape and size of the modern horse.

4.1.2 Breeding

Selective breeding, or artificial selection, is when pairs of animals are intentionally bred together in order to propagate some desirable trait which they both share to their offspring.

Humans have been selectively breeding horses for hundreds, if not thousands, of years and there are over 300 breeds of horse in existence today. The aim of breeding has always been to improve the performance of a particular animal at some task, staying true to the phrase "form follows function".

Over the years horses have been bred to display specific combinations of traits such as speed, acceleration, manoeuvrability, endurance, strength, disposition, ridability, stature and aesthetics. Some of these

traits complimented each other whilst others are more mutually exclusive. A large, muscular horse for example, may exhibit great strength but this may be at the expense of speed. If the animal is to be used to pull a plough through a field, great speed is not a necessary trait and the compromise is justifiable. Conversely, a race horse sacrifices strength for a lighter, faster body which allows it to excel at its function; winning the race.

Particular breeds of horse are classified as a certain type based on form, function and temperament. Draught horses are large animals usually bred for agricultural labour. A packhorse is generally a smaller horse or pony, used for carrying items on its back, often over rough terrain. Sporthorses are specifically bred for sporting events such as dressage, show jumping and carriage driving. Hot-blooded breeds, referring to temperament, are bred for speed and agility and include the highly popular Arabian horse and Thoroughbred, used for horse racing.

These different types of horse can often be visually identified by their size and shape [86]. The painting shown in Figure 4.2 depicts a scene containing several breeds of horse. The Belgian, Noriker and Clydesdale are all draught horses originally from Belgium, Austria and Scotland respectively. These strongly built workhorses are still used to this day for pulling loads and other agricultural tasks.

The Oldenburger is an example of a modern Sporthorse whose middleweight body is considered particularly suitable for show jumping and dressage. The much smaller Dales Pony possesses great stamina. Originating from Northern England, this horse has traditionally been used as a pack and artillery animal.

Figure 4.2: "Pferderassen II" by the 19th Century German master horse painter Emil Volkers. Image depicts a variety of horse breeds, clockwise from top-left: Neapolitan, Oldenburger, Noriker, Clydesdale, Dales Pony, Belgian.

The final horse is the Italian Neapolitan. Similar to modern-day Hot-blooded breeds, they were historically bred by nobles for transportation and cavalry. Although a revered horse of the middle ages, the Neapolitan horse became extinct in 1950 after nearly a century of decline.

When compared to each other as in Figure 4.2, the different types of horse are quite distinct. Visually, the size of the animal in terms of height and width may give an observer an insight into its functionality. Within each classification of animal however, more subtle differences exist between breed. Length of limb, inclination of a particular bone, shape of the back and arch of the neck are some of the more obvious differences between breeds of horse of the same type.

For animation purposes, it is imperative that the animal model used reflects the task it is being used for. For example, a heavy Draught horse galloping in a race and jumping over fences may not be believable to an audience. Similarly, animations of a scene set in a specific time period should utilise models that reflect the breeds of horse that were available at the time.

There are many differences between breeds of horse that are of little concern to an animator of course. Variations in the number of vertebrae and some of the smaller bones are often of little consequence to the animal's function, however, some seemingly small differences can be an intentional product of selective breeding for a particular task.

Within a particular breed, there will also be variation between animals but there is usually a particular set of characteristics that a breeder strives to achieve, known as the optimal conformation.

Conformation

The optimal conformation of a horse is the set of characteristics relating to bone structure, musculature and body proportions which should allow the animal to perform highly at its task whilst preventing injury [84].

Studies of animal conformation can provide skeletal dimensional data which can be utilised for computer model creation [92]. These studies can also provide information about how a particular conformation can affect movement. For example, an understanding of how variations in length and inclination of a particular bone affect an animal's motion is of interest to a breeder attempting to produce a high-class race horse, and an animator attempting to produce a realistic animation.

A list of general horse conformation guidelines is presented in Appendix Section A.2. Information such as this can be used to ensure that certain aspects of a computer constructed animal are in keeping with the real-life animal equivalent.

Continuing on the topic of utilising natural observations for realistic animation, in the following section, the anatomy of a horse is discussed.

4.2 Anatomy

Anatomy is the scientific study of bodily structure [28] and knowledge of an animal's internal and external composition can be used to increase the realism of animal animations [203].

The specific focus of this thesis is cursorial quadrupedal animal animation. As stated previously, a quadruped is an animal (or robot) that uses four legs for locomotion and a cursorial quadruped has limb structure adapted for running.

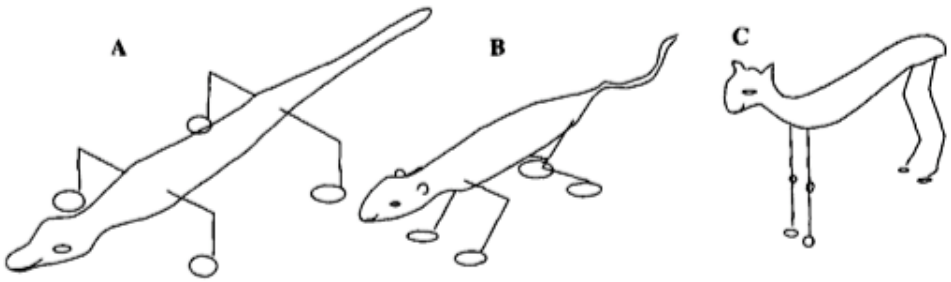


Figure 4.3: Quadruped types. **A** reptilian, **B** non-cursorial, **C** cursorial. (Image taken from [126])

In Figure 4.3, taken from [126], three different classes of quadrupeds are shown; reptilian, non-cursorial and cursorial. The important morpho-

logical difference to note in these images in terms of locomotion is the positioning and structure of the limbs in relation to the animal's body.

Non-cursorial mammals comprise rodents and small carnivores [11]. These mammals tend to weigh under 3kg and run with the humerus and femur in a near horizontal position. Examples of non-cursorial mammals include ferrets, rats, jirds and coypus (unusually large non-cursorial mammals) amongst many others.

Cursorial mammals, such as ungulates and large carnivores weighing more than 3kg, have much straighter legs with the femur and often the humerus nearer to vertical. Examples of cursorial mammals include rhinosceroses, horses, camels, wildebeest, sheep, dogs and cats (unusually small cursorial mammals).

The research and experiments presented in the following chapters focus specifically on the horse. As horses are dynamically similar to other cursorial quadrupeds (discussed in Section 4.5), each horse-specific animation experiment is applicable to the other cursorial quadrupeds, assuming the availability of skeletal measurements and other anatomical information. Anatomical and behavioural idiosyncrasies of a particular species, for example the spinal flexibility of cats (felidae), can be accommodated on a case by case basis.

The horse is chosen for study due to its continued presence in human society; horses are still used for work, sport and entertainment purposes [19]. Because of this, much research and data is available on the subject of horse anatomy. In the following sections, the locations of certain equine anatomical features are discussed, often using special terminology. A brief list of these terms is presented in Appendix Section A.3.

The horse, as with the vast majority of animals, has a body composed of multiple distinct tissues. Each tissue has a specific function and those that are particularly significant in the locomotion system are discussed in the following section.

4.2.1 Tissues involved in locomotion

A tissue is a collection of specialised cells and products of these cells [28].

An animal's body comprises many biological systems. Each of these systems is a set of organs that work together to carry out some function. Each of these component organs performs a specific task and is made up of a collection of tissues. There are four basic types of tissue in an animal: nervous, muscle, connective and epithelial. In terms of animal locomotion, aspects of the muscle and connective tissues are of interest.

Muscles are the contractile tissues which produce force and, in the case of skeletal muscle, enable locomotion. The muscle tissue is called contractile as it can contract on demand, either consciously or unconsciously, as well as involuntarily. Muscles tend to be arranged in opposition; as a particular muscle group contracts, an opposing muscle group relaxes. The force that a particular muscle produces is proportional to its cross-sectional area at its thickest point and the velocity at which a muscle contracts is proportional to the muscle fibre length. While muscles can dynamically contract, they are distinct from connective tissues which stretch and contract passively.

Connective tissues are a type of fibrous tissue mainly comprising collagen, a group of natural proteins. Connective tissue can stretch and contract and its main function is to hold other tissues together. There

are six main types of connective tissues: loose connective tissue, adipose tissue, blood, collagen, cartilage and bone. In terms of biomechanics, the tendons, ligaments, cartilage and bone are the most important.

Tendons which connect muscle to bone are made of regular collagen; it is called regular as the fibres are lined up in parallel. Tendons are able to withstand tension and also exhibit certain springlike elastic properties that passively improve stability during locomotion [5].

Ligaments which connect bone to bone are also made of regular collagen. They are similar to tendons, however, while they exhibit some elasticity, they lack the springlike properties. The ligaments play an important role in constraining joint rotation and maintaining body shape and structure.

Cartilage is also composed of collagen, except the fibres are irregularly positioned. The strength and flexibility of cartilage allow it to absorb shock and it is found between vertebrae and at the end of bones in some joints, amongst other places.

The final connective tissue of interest to us are the bones. Also known as osseous tissue, bones are a type of hard, rigid connective tissue. Besides being used for support, movement and protection, bones also store minerals and produce red and white blood cells. The structure of a bone means that it is lightweight but relatively strong and hard.

Bones come in many different shapes and sizes and the following section takes a more detailed look at bones and the joints between them.

4.2.2 Bones and joints

The bones present in a horse's body are classified into the following categories: longs bones, short bones, flat bones, irregular bones and sesamoids [77]. The long bones act as levers used in locomotion and are mainly found in the limbs. The short bones absorb concussion in the joints. Flat bones, such as the skull, pelvis and ribs protect the vital organs whilst the irregular bones of the spinal column protect the spinal cord. Finally, the sesamoids are bones enclosed within a tendon.

Bones are attached to each other via ligaments and the location at which two or more bones meet is called a joint. Joints vary in the degree of movement they allow and are classified as follows:

- synarthrosis - little or no movement, e.g. in the skull
- amphiarthrosis - very small movement, e.g. between the vertebrae
- diarthrosis (synovial) - large range of movements, e.g. the shoulder, hip, elbow, knee

The joints of the skull, called sutures, are fibrous synarthrosis joints that allow no movement and over time become bone [157]. The amphiarthrosis joints connecting the vertebrae are made up entirely of cartilage, allowing very little movement, which leads to inflexibility in the horse's back.

In contrast, the diarthrosis joints allow for much freer rotation between connected bones. The ends of the bones that meet at these joints are covered in cartilage allowing the bones to slip past each other during movement whilst synovial fluid, secreted by the membrane covering the cartilage, provides further lubrication [157].

The freedom of movement allowed by a particular joint depends on the type of joint it is and its morphology. A joint's motion is usually described using the terminology listed below:

- Flexion: joint-angle decreases along the sagittal plane, i.e. forward to backward
- Extension: joint-angle increases along the sagittal plane, i.e. backward to forward
- Abduction: movement away from the body's midline
- Adduction: movement towards the body's midline
- Rotation: circular movement around a central point or axis

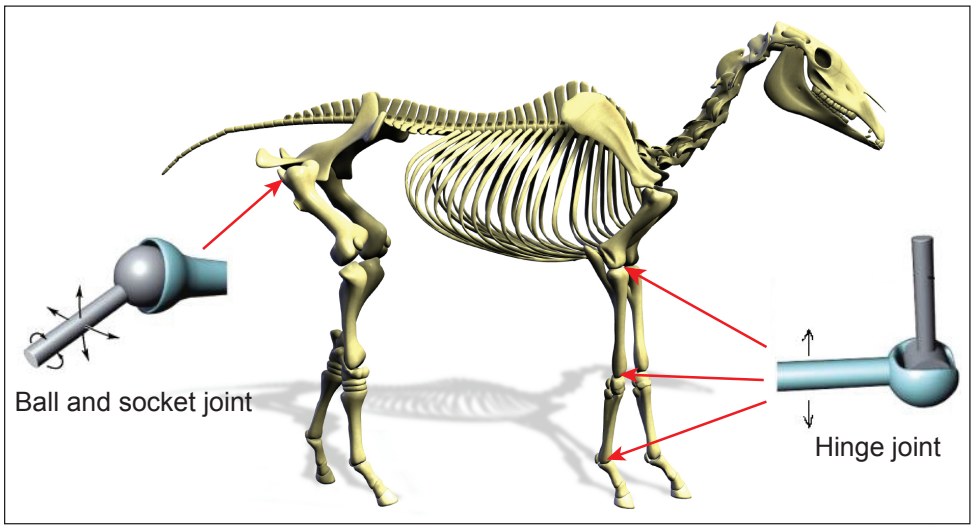


Figure 4.4: Two common joint types found in a horse's body. (Inset joint images © Pearson Education, Inc. 2007)

Using this terminology, the degrees of freedom allowed by the two most common types of joint, shown in Figure 4.4, can be described. Although the horse's body contains many complex joints, in a simplified model,

the joints that exhibit significant movement can be described as being either a hinge joint or ball and socket joint.

A hinge joint is analogous to a hinge in a door; it allows flexion and extension about a single axis. In both humans and horses, the elbows and knees are hinge joints.

A ball and socket joint is much more flexible. It allows flexion, extension, abduction, adduction and rotation. In both humans and horses, the shoulder and hip are examples of a ball and socket joint, as will be discussed in the following section on the equine musculoskeletal system.

4.3 Equine musculoskeletal system

When producing realistic animations of horses in motion, the most important aspect of equine anatomy is the musculoskeletal system.

The musculoskeletal system is the system of organs that enables an animal to move [4]. Also known as the locomotor system, it comprises the skeleton (bones and joints), ligaments, cartilage, muscles, tendons and other connective tissues. As well as enabling an animal to support, stabilise and move itself, the musculoskeletal system gives the animal its recognisable shape.

The horse skeleton is made up of 205 bones on average [81] and each body segment of the horse contains a combination of the bones introduced in Section 4.2.2.

These distinct sections of a horse's body are described individually in the following sections.

4.3.1 Neck and head

The neck is made up of seven cervical vertebrae as shown in Figure 4.5. Counting from the distal end of the neck, to which the skull is attached, the first and second vertebrae are known as the atlas and the axis respectively. These vertebrae are anatomically different from the other cervical vertebrae as they allow a large degree of movement. The atlas allows the horse the rotate its head up and down whilst the axis allows side to side motion.

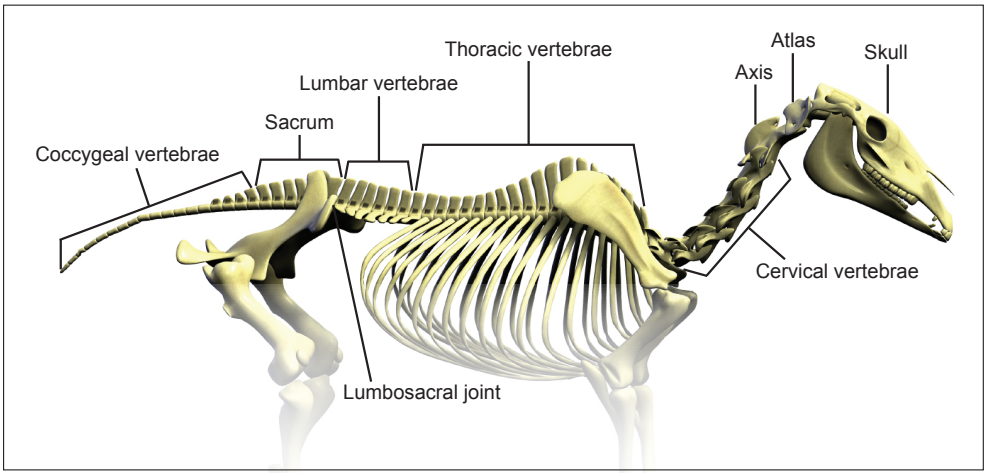


Figure 4.5: Detailed diagram showing the bones of a horse's neck, back and tail.

The other five cervical vertebrae allow the animal to move its neck from side to side, stretch and arch it. Movements of the neck can significantly affect the horse's balance and centre of gravity. For instance, lowering the neck puts more weight on the forelimbs and brings the centre of gravity forward whilst raising the neck increases the load on the hindlimbs and shifts the centre of gravity backwards [81]. Proceeding in a caudal direction (towards the tail), the motion of the vertebrae becomes limited, giving rise to significant stiffness in the back.

4.3.2 Back and tail

The back is the section of spine running from the neck to the tail as shown in Figure 4.5. It includes eighteen thoracic vertebrae with almost no flexibility, six lumbar vertebrae with some flexibility relative to the thoracic vertebrae and five fused sacral vertebrae with no movement.

Unlike the cervical vertebrae, the vertebrae of the back allow for only minor movements and this is usually due to hindlimb motion [157]. The stiffness of the spine allows it to support the thorax and abdomen as well as protecting the spinal column.

The five sacral vertebrae are collectively called the sacrum and forms part of the pelvis; the link between the trunk and the hindlimbs. The lumbosacral joint is a synovial joint between the sixth lumbar vertebra and the first sacral vertebra. Although it has limited flexibility, it allows the pelvis to rotate forward under the body during fast gaits like the canter and gallop, as will be discussed in Section 4.4.2.

The tail comprises eighteen coccygeal vertebrae, decreasing in size from the first to the last. The first coccygeal vertebra connects directly from the last sacral vertebra. Like the neck, the tail is very flexible. Besides being used as a fly swatter, the tail positioning changes as the horse turns to provide some balance. The tail is also an indicator of the animal's emotion.

4.3.3 Forelimb

The bones of a horse's forelimb are shown in Figure 4.6. As discussed in Section 4.1.1, a horse has evolved for speed and as such, the limbs are

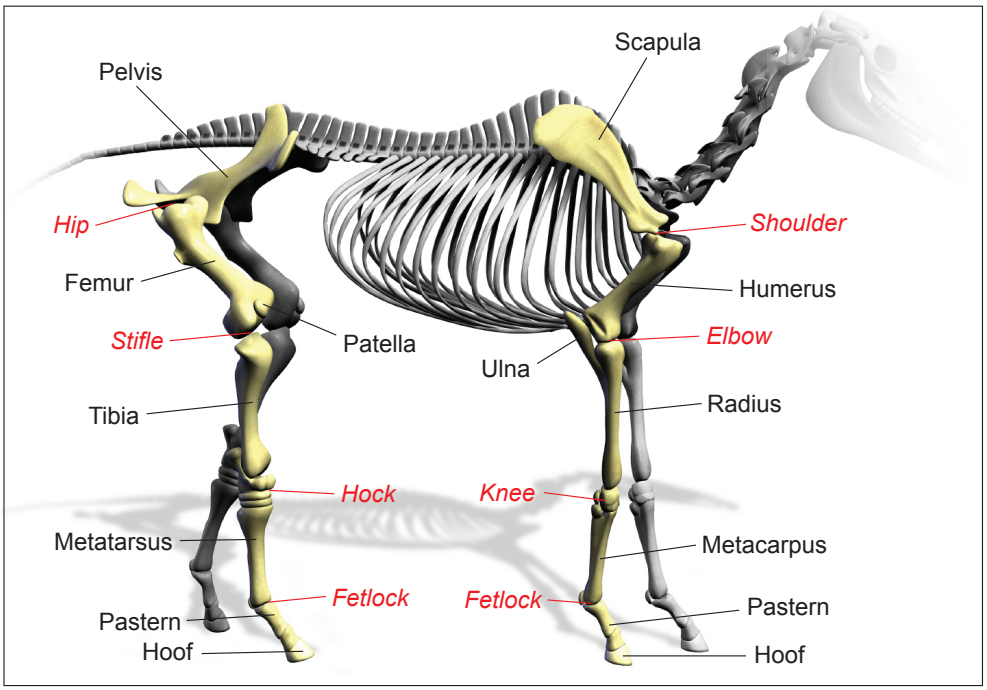


Figure 4.6: Detailed diagram showing the bones of a horse's forelimb (right) and hindlimb (left). Bone names are shown in black and joint names are shown in italicised red.

long and the weight of the lower limb is minimised as there is no muscle below the knee. Motion in the lower limb is produced as contractions of muscles higher up in the leg are transmitted via long tendons.

Another interesting feature of the forelimb is that it is only attached to the body by muscles and ligaments. This design allows the forelimbs to be used as shock absorbers, as during locomotion they withstand large impacts.

Starting from the top of the forelimb, the scapula is a flattened triangular shaped bone with a typical inclination of 45°. This angulation coupled with the bone's length influence the length of stride. Unlike humans where the scapula is connected to the body via the collar bone (clavicle), the horse's scapula does not have any connection via bone.

Instead the horse's thorax, the region of the trunk from the neck to the diaphragm, is slung between the scapulae by the thoracic sling, comprising muscles, tendons and ligaments.

The humerus is connected to the scapula by a ball and socket joint called the shoulder. The humerus is angled for shock absorbtion, ideally at 60° to the horizontal. The movement of the shoulder is mainly flexion and extension, however, some abduction, adduction and rotation also occurs. The humerus attaches to the radius and ulna at the elbow joint.

The elbow is a hinge joint and only allows flexion and extension along the sagittal plane. The radius and ulna are fused together in a horse's forelimb; the ulna is small relative to the longer radius.

The radius connects to the metacarpal bones at the knee or carpus. The knee comprises seven or eight small carpal bones which allow it to act as a shock absorber. Like the elbow, the knee is a hinge joint permitting flexion and extension along the sagittal plane.

Below the knee, the metacarpal bones comprise two (vestigial) splint bones and the load-bearing cannon bone. The cannon bone is connected to the long pastern bone and two sesamoid bones at the fetlock joint which is a hinge joint with flexion and extension motion along the sagittal plane.

The lower bones of the fetlock joint, known collectively as the pastern, comprises the long (first) and short (second) pastern bones. The angle of these bones should be about the same as the scapula, around 45°. There is a joint known as the pastern joint between the long and the short pastern. Although it is a hinge joint, its movement is very limited.

The aforementioned sesamoid bones are located at the back of the fetlock joint and behave as pulleys as the pastern bones are moved by tendons from the muscles of the forearm (near the radius).

The final major joint of the forelimb is the coffin joint; a hinge joint between the short (second) pastern bone and the pedal and distal sesamoid (navicular) bones within the hoof. The distal sesamoid again behaves as a pulley for tendons stemming from muscles farther up the leg, however, the coffin joint allows very limited flexion and extension.

4.3.4 Hindlimb

Whilst the forelimb acts to absorb shock during locomotion, the hindlimb provides the propulsion.

The power and speed of a horse comes from the large muscles of its hindquarters [19]. These muscles extend from the sacral and coccygeal vertebrae to the stifle joint. Whilst the lower hindlimb (below the hock) is anatomically the same as the lower forelimb (below the knee), the upper hindlimb is significantly different as can be seen in Figure 4.6.

The pelvic girdle (pelvis) includes the sacrum and the first three coccygeal vertebrae introduced in Section 4.3.2. It also includes the two pelvic bones (os coxae) which form part of the hindlimbs. The pelvis and the femur meet at a concave surface on the pelvic bone called the acetabulum, forming the hip joint.

The hip is a ball and socket joint allowing a large range of movement. The femur which attaches to the pelvis at the hip is a long, strong bone to which the muscles of the hindquarters connect, providing the animal's power.

The femur connects to the lower bones via the stifle; a large joint whose main function is to rigidify the hindlimb upon contact with the ground. A sesamoid bone called the patella is located at the stifle and by contracting certain muscles, the patella is upwardly fixated and prevents flexion of the stifle.

The stifle and the hock, described below, share reciprocal action which means that as one flexes or extends, so does the other. During the upward fixation of the patella, this reciprocal relationship prevents flexion in the hock as well as the stifle, however, joints lower in the limb can still be flexed.

The stifle joint is actually a complex collection of three joints, however, for simplicity it can be viewed as a single hinge joint with a large degree of freedom along the sagittal plane. Between the stifle and the hock runs the long tibia bone. Another bone called the fibula is also present but its very small size render it essentially vestigial.

At the distal end of the tibia is the hock joint. Although comprising numerous tarsal bones, the hock has a smaller range of movement than the knee joint in the forelimb; it absorbs shock by maintaining a constant semi-flexed state. Below the hock, the limb is anatomically the same as the forelimb as described in Section 4.3.3.

The skeletal features described here pertain to a fully-grown adult horse. Whilst the skeletal structure is largely the same regardless of age, there are some differences. For animation purposes, only those differences which alter the animal's aesthetic appearance and its locomotion are of interest. These changes of structure with age are briefly discussed in the following section.

4.3.5 Skeletal allometry

In biology, allometry is the study of how body parts grow at different rates, giving rise to differently proportioned bodies depending on an animal's age [170].

A movie or video game will often require a scene comprising multiple horses of different ages. An animator may be tempted to take a single animal model and uniformly scale the entire model to indicate a younger or older animal. This approach is ultimately incorrect as generally, animals are not born with the same skeletal proportions as their fully-grown counterpart. To produce a model with proportions correct to its age, observations of skeletal allometry can be used.

A mammal's skeleton does not scale isometrically with body mass. As the body mass of the mammal increases, the skeleton becomes more and more massive relative to the size of the body. An in-depth discussion of allometry is beyond the scope of this thesis, however, there are numerous published studies which provide valuable data to an animator; a large set of data for terrestrial mammals relating total body mass to the mass of the animal's long bones is available in the literature [35]. Further data on how bone dimensions scale with body mass is also available [66, 10, 160].

Allometric studies of mammals are a great source of data when one is constructing an animal model and have helped researchers to estimate the body mass of extinct animals [51]. Studies of allometry also extend beyond skeletons and can be used to predict the physiological properties of other organs [170].

Equine allometry

Horses are a widely studied animal and growth-rate data is available through many published studies [188, 74, 99].

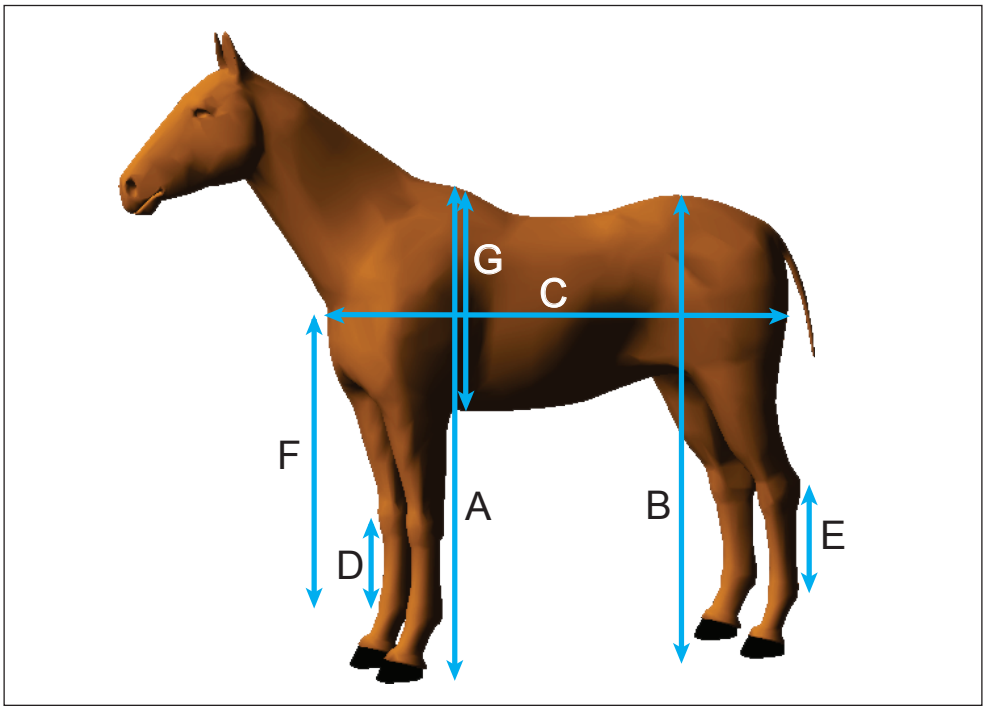


Figure 4.7: Measurement points for analysing growth of horses [188].

Attempting to measure the length of every bone in an animal's body in vivo is not possible so instead measurements are taken between various external points on the body, as displayed in Figure 4.7. Distances measured include wither height (**A**), hip-height (**B**), body length (**C**), knee to pastern length (**D**), hock to pastern length (**E**), point of shoulder to pastern length (**F**) and depth of girth (**G**). These point-to-point distances tend to be standard when assessing the allometry of a horse and other animals.

In general, the skeletal segments of animals of the same breed follow a distinct growth pattern, with deviations usually occurring due to abnormal feeding or environmental factors. Animals of differing breeds however, will grow at different rates and have a broad weight range, as illustrated in Figure 4.8.

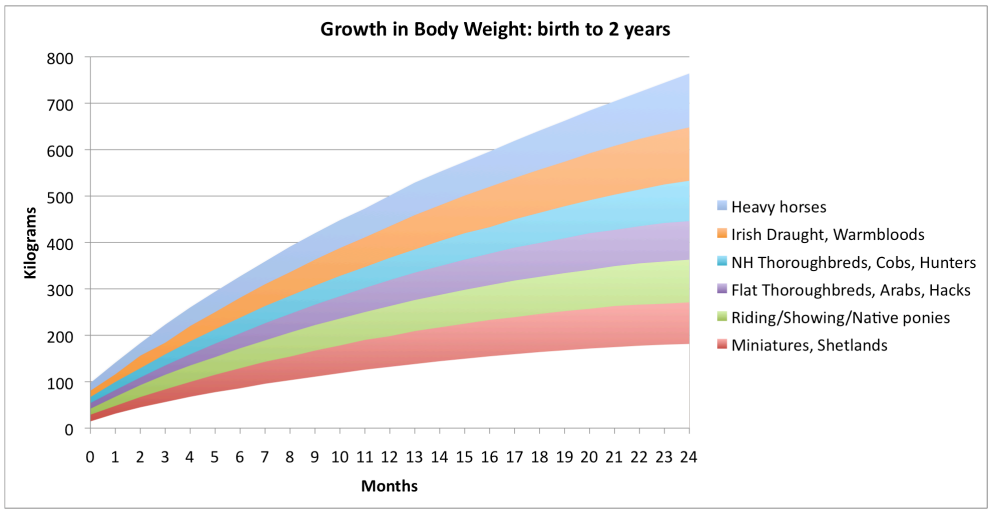


Figure 4.8: Growth in body weight from birth to 2 years of age. Growth-rate and body mass is shown for multiple breeds [59]

.

Allometric data can be used to improve the accuracy and aesthetic realism of a horse animation, provided that data pertaining to the relevant breed of horse is available. If growth or dimensional data for a specific breed or species of animal is not available, values can be approximated from images and video. The many published studies mentioned previously in this section could also be used to estimate an animal's proportions based on body mass.

In Chapter 9 of this thesis, experiments that make use of allometric data for horses are described, as the issue of how the proportions of a horse affect its movement is explored.

In preparation for this and the other experiments concerning equine motion, the following section describes the basic motion patterns that all horses use for locomotion, regardless of breed and age.

4.4 Movement

The horse is a terrestrial mammal and like the majority of land animals, it uses its legs to move from one place to another.

Like most walking animals, horses are quadrupeds and during locomotion, muscles contract periodically to produce movement in its limbs. The motion of the limbs cause the feet, or hooves, to push off the ground surface, thrusting the animal forwards.

The sequence in which the limbs move is pivotal in the locomotion process and is referred to as the animal's gait.

4.4.1 Gait patterns

A gait is a pattern in which the limbs of an animal move during locomotion [90]. Gait patterns describe the sequence and timing in which each limb pushes off the ground as an animal moves over a solid substrate. Animals move in a rhythmical manner and as such, gait patterns are cyclical in nature; a gait cycle begins when a foot contacts the ground and ends when that same foot contacts the ground again.

During a gait cycle, each limb will experience a stance phase and a swing phase. A limb is in the stance phase when its foot or hoof is in contact with the ground. The duration of this contact (as a fraction of a full cycle) is known as the limb's duty factor. The swing phase

commences when the limb breaks contact with the ground and is swung forward in preparation for its next grounding.

Gaits are usually classified as being either a walking or running gait. In walking gaits there is always at least one limb in contact with the ground. Conversely running gaits have at least one period of suspension in which no limbs are in contact with the ground. The running gaits are naturally faster than the walking gaits and for legged animals, velocity is often described in terms of stride length.

Not to be confused with step length, stride length is the distance between two successive placements of the same foot during locomotion and is calculated using Equation 4.1.

$$\text{stride length} = \text{velocity} \;/\; \text{stride frequency} \tag{4.1}$$

The relationship between stride length and the duration of the stance and suspension phases depend on the gait. Typically, as velocity increases within a particular gait, the duration of suspension also increases.

In general, a particular gait is efficient over a range of velocities. The animal can increase or decrease its velocity within that range without changing (transitioning) to a new gait. Whilst an animal can change its velocity by simply increasing its stride frequency, velocity can also be changed through adjustment of its limb extent.

Limb extent is defined here as the distance between the farthest grounded hoof position attained in the forward and backward directions along the sagittal plane over a single gait cycle.

An animal can reduce its velocity by extending its limbs less, and therefore taking shorter steps, while maintaining a constant rate of steps per second. Conversely, velocity increases if the animal reaches farther forward and backward during each step.

All legged animals move with a specific set of gaits which differ among species. In the following section, the gaits used naturally by the horse are presented.

4.4.2 Equine natural gaits

A horse has four natural gaits. In order of increasing speed, they are the walk, trot, canter and gallop.

The footfall sequences of the horse's natural gaits are presented in Figure 4.9. For each of the gaits, an illustration of a horse is shown with values marked adjacent to its limbs. These values indicate the phase difference between the limbs as a percentage of a full gait cycle. The presented values are generally consistent among all horses, with some small variations [19].

The stick diagrams to the bottom of Figure 4.9 graphically display the footfall sequences for each gait with the black bars indicating a limb's stance phase. From this diagram it can also be seen that gaits are described as being either symmetrical or asymmetrical.

The walk and trot are symmetrical gaits meaning a left and right pair of limbs move alternately to each other. Gaits such as the canter and gallop are classified as asymmetrical, as pairs of limbs move together.

As a pair of limbs is moving during asymmetrical motion, the limb whose hoof touches the ground temporally after and positionally in front

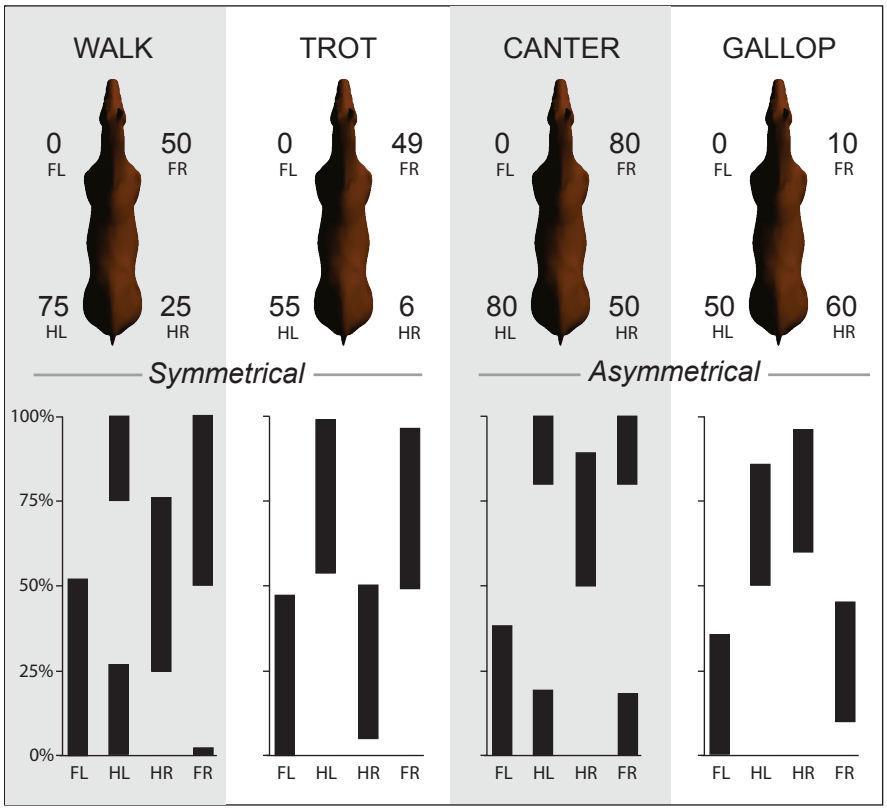


Figure 4.9: Top: typical timing of footfalls as a % of a full gait cycle, for the natural gaits (FL: fore-left, FR: fore-right, HL: hind-left, HR: hind-right). Bottom: hoof contact with the ground (thick black bars indicate contact).

of its partner is referred to as the "leading" leg. The other limb of the pair is known as the "trailing" leg. The horse can change which leg leads to affect speed and improve balance, especially when turning.

An example of "changing the lead" in the asymmetrical gaits is shown in Figure 4.10. The diagram graphically displays the difference between the transverse and rotary canter/gallop as the lead is changed from the right forelimb to the left forelimb. Horses usually use the transverse canter and gallop except in horse racing, when use of a rotary gallop can be approximately five miles an hour faster [173].

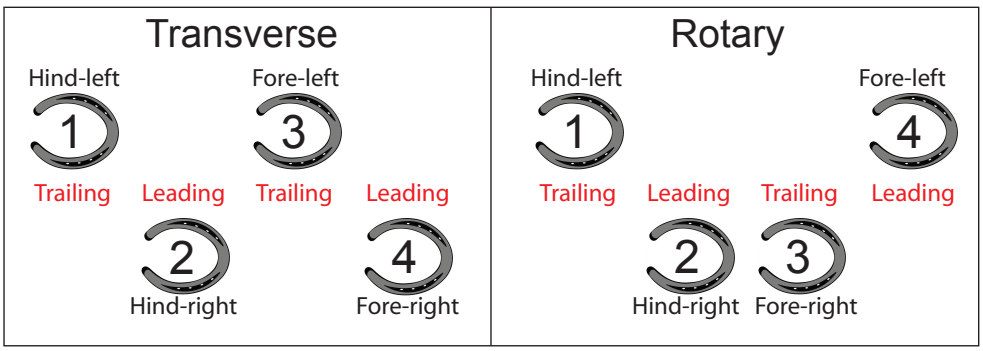


Figure 4.10: Variations of the canter and the gallop by changing the leading foot (forelimb). The numbers indicate the sequence of the footfalls.

The transverse and rotary versions of the canter and gallop are considered variations of the same gait despite having different footfall patterns. In the following subsection, other variations of the natural gaits are briefly discussed.

Gait variations

A horse in motion may move with variations of the natural gaits and some specific breeds use gaits that are only natural to them.

Often variations of a gait are defined by how the limbs are placed in relation to the body during locomotion. Limb placement affects the animal's centre of gravity and its limb extent which in turn affects its stride length [36].

The natural form of a gait with normal limb extent and stride length is often referred to as the "working" form of the gait; the limbs are not greatly extended allowing the animal to move in a comfortable, sustainable manner.

A "collected" gait is one in which the horse carries more weight on its hind legs and draws its body in on itself. It does this through down-

ward flexion of the lumbosacral joint, which is called "engagement of the hindquarters". Collection is a natural response to danger but can also be observed in any gait during locomotion. Collection also affects velocity as the strides of a collected gait tend to be relatively short.

An "extended" gait in contrast employs large limb extensions and thus the strides are longer and made with great impulsion. As a consequence, extended versions of the faster gaits can include lengthy periods of suspension.

A "medium" gait lies somewhere between the working and the extended gait in terms of stride length and impulsion. A horse's gait can also be described as being "disunited" if the footfall sequence is irregular or not matching the pattern that is normally expected for a particular gait.

The interaction between humans and horses has also introduced a variety of new gaits and gait variations. In dressage, some breeds of horse are trained to use unnatural gaits, such as the passage and piaffe.

Other breeds of horse demonstrate idiosyncratic ambling gaits from birth, such as the Tölt of the Icelandic horse, the Paso of the Peruvian horse, the Rack of the American Saddlebred and the Fox-trot of the Missouri Foxtrotter.

Further information about horse gaits is presented in Appendix Section A.4, with a list of the most common gaits of the horse and their characteristics provided in Table A.1. The value ranges for characteristics of the common equine gaits are also presented in Table A.2.

The gait employed by a horse in motion ultimately depends on the velocity at which the animal wishes to travel. If a change in velocity is

required, the horse can either alter its limb extent and stride frequency within its current gait, or it can transition to a different gait. The subject of gait transitions is discussed in Section 4.4.4. Before this however, the influence of neck and back movements on balance during locomotion is briefly discussed.

4.4.3 Neck and back

While a gait is often characterised by its leg movement, each natural gait has characteristic neck motions which affect balance and stride length.

The neck has an important role to play in locomotion. A horse can shift its centre of gravity by moving its head and neck, significantly affecting weight distribution and balance. By lowering the head, the weight shifts forward towards the forelimbs. Conversely, raising the head shifts the centre of gravity back towards the hindlimbs [157].

During locomotion, the raising, lowering, flexing and extending of the neck are collectively referred to as "balancing gestures". The scale and manner of influence that these gestures have on an animal's balance and gait depends on the morphology and conformation of the neck (discussed in Appendix Section A.2). As well as influencing balance, the Brachiocephalius muscle of the neck which runs from the atlas vertebra (close to the head) to the humerus is largely responsible for pulling the forelimb forward during locomotion.

The combination of these factors results in distinctive neck motions, depending on the gait the horse is moving with [81]. A detailed description of the neck and back motions for the natural gaits is provided in Appendix Section A.5.

4.4.4 Transitions

Transitions between gaits occur when an animal changes its velocity during locomotion.

Although a horse can increase or decrease its velocity by increasing or decreasing its limb extent, a horse will often transition up or down a gait rather than adopt extremely long or short steps.

When free, a horse tends to gradually change between adjacent gaits as its velocity shifts outside its current gait's range, e.g. halt to walk, walk to trot, trot to canter etc. In some cases, transitions between non-adjacent gaits occur; a startled horse may directly transition from a walk to a gallop for example.

Technically, a change in velocity, or foot placements for balance, within a particular gait is also referred to as a transition but for this discussion, a transition refers to a change from one gait to another, and to or from a halt.

As gaits are distinguished by their footfall sequences, the transition between gaits involves adjustments of the phase difference between limbs from one value to another. As the transition is occurring, the horse either increases or decreases its limbs' rate of movement to achieve the target phase difference.

This change of phase can take place over one or more gait cycles. If the transition takes place over multiple gait cycles, some intermediary steps may be taken to incrementally change the limb phase difference to that of the new gait.

The footfall sequence of a transition is often idiosyncratic to a particular animal and can be inconsistent. As such there are no set transition patterns or number of cycles over which a particular transition occurs. It is also difficult to visually assess how a transition occurs without specialist equipment.

In general, the pattern of change and time taken to complete a transition varies hugely depending on factors such as current gait and velocity, target gait and velocity, terrain, breed or habit, however, footfall sequences for gait transitions have been measured [118, 24, 50, 14, 15].

Cause of transitions

The question of why transitions occur at particular velocities has been extensively studied in the biology and biomechanics field.

In some studies, experiments have found that the trot-gallop transition occurs when the peak ground reaction force reaches a critical level of between 1 to 1.25 times the animal's body weight [57]. It was also found that increasing the load on the horse reduces the speed at which the transitions occurred.

Human gaits are examined to ascertain whether transitions occur to maintain metabolic efficiency, to reduce ground reaction force or due to simple mechanical factors [166]. After extensive testing of the walk-to-run and run-to-walk transitions, it was concluded that humans in motion transition between gaits to bring ground reaction forces and skeletal loading below some critical force level.

Other studies suggest that the speed at which transitions occur is related to an optimal metabolic cost of running [94]. The hypothesis

that the walk-to-trot transition in horses and other animals is triggered by the dynamics of an inverted pendulum system, and occurs to maximise metabolic economy, has been validated [75].

A similar study investigating the effect of reduced gravity on the walk-to-run transition speed of humans also supports this hypothesis [107]. Using a minimal mathematical model of a biped, the traditional "inverted pendulum walking" and "impulsive running" were found to minimise work when moving at slow and fast speeds respectively [183].

The implications of this hypothesis and the observation that the velocity at which transitions occur is relatively equal between many species of mammal is discussed next in the final section of this biology chapter.

4.5 Dynamic similarity

Dynamic similarity theory postulates that different mammals move in a dynamically similar manner whenever they travel at speeds that give them equal values of the dimensionless Froude number [9].

Dynamic similarity theory as summarised above relates to land mammals. In this thesis, unless stated otherwise, all references to an "animal" in relation to dynamic similarity theory refer specifically to cursorial quadruped mammals.

Essentially, dynamic similarity theory allows one to predict the gait characteristics an animal will exhibit when travelling at a particular velocity. The predictions depend on the availability of gait data measured from another animal; this data can often be found in the biology literature [9].

The Froude number upon which dynamic similarity is based relates velocity, gravity and an animal's hip-height (the distance from the ground to the hip). It is calculated by Equation 4.2.

$$Fr = v^2/gh \tag{4.2}$$

where Fr is the Froude number, v is velocity, g is acceleration due to gravity and h is height of the animal's hip from the ground at stance.

The Froude number originated in the naval architecture field, where it was used to quantify the resistance of a ship's hull travelling through water. The Froude number and its development from shipbuilding to gait analysis is documented in the literature [199].

Dynamic similarity theory is based on the observation that animals moving at equal Froude numbers have similar gaits, i.e. they exhibit the following characteristics:

- Limbs move in the same phase relationship
- Relative stride lengths are equal
- Feet have equal duty factors
- Feet exert comparable forces on the running surface
- Costs of transport are relatively equal

In this thesis we are mostly interested in the visible aspects of locomotion. Issues such as cost of transport are of less concern. The following section presents details of how dynamic similarity theory is used to predict gait characteristics which can be used in animal animation systems.

4.5.1 Predictions from dynamic similarity

Whilst one cannot determine the properties of a gait from an animal's Froude number alone, if gait characteristics for another animal travelling at that particular Froude number are known, then those characteristics should hold for all other animals travelling at the same number.

Table 4.1: Gaits and corresponding Froude numbers.

Gait	Walk	Trot	Canter	Gallop
Froude	0.0 - 1.5	1.51 - 2.5	2.51 - 3.5	3.51 - 4.5
Symmetry	symmetrical	symmetrical	asymmetrical	asymmetrical

Table 4.1 presents the natural gaits of the horse and corresponding Froude number ranges. These values are based upon measurements taken from large numbers of different cursorial quadrupeds [9].

Table 4.2: Dynamic similarity power law equations.

Prediction	Symmetrical	Asymmetrical
Fore duty factor	y = 0.52 * $Fr^{-0.14}$	y = 0.52 * $Fr^{-0.28}$
Hind duty factor	y = 0.51 * $Fr^{-0.18}$	y = 0.53 * $Fr^{-0.28}$
Relative stride len.	y = 2.4 * $Fr^{0.34}$	y = 1.9 * $Fr^{0.40}$

The power law equations shown in Table 4.2 are also taken from the literature [9]. From these equations, the relative stride length and forelimb/hindlimb duty factor values can be calculated for both the symmetrical and asymmetrical gaits.

To predict a value for a particular Froude number, the Froude number is substituted for Fr in the relevant equation and y refers to the calculated value. Plots of these values for a range of Froude numbers are presented in Figure 4.11. The values in this figure are calculated for a cursorial quadruped (horse) with a hip-height of 1.24524 metres.

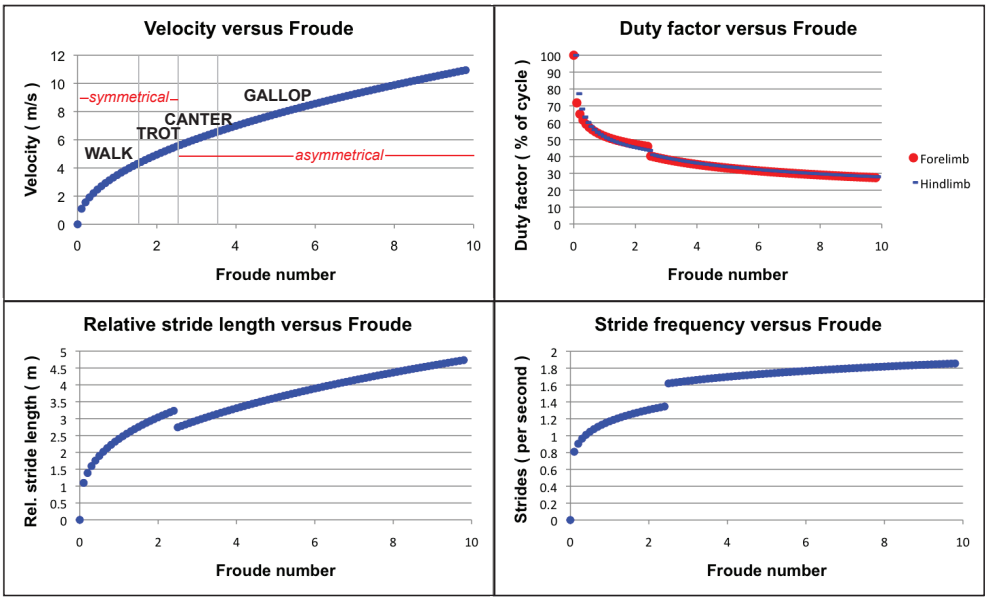


Figure 4.11: Predictions of dynamic similarity theory for a cursorial quadruped (horse) with hip-height of 1.24524m.: *Top-left* Velocity versus Froude number as predicted by the Froude equation. The predicted gaits, transition points and gait symmetries are also displayed. *Top-right* Duty factor versus Froude number. *Bottom-left* Relative stride length versus Froude number. *Bottom-right* Stride frequency versus Froude number.

In the top-left image of Figure 4.11, velocity is plotted annotated with the predicted gaits shown in Table 4.1. The velocities are calculated using Equation 4.2 and the Froude range for the gaits is from the literature [9].

In the top-right of Figure 4.11, duty factor values for both forelimb and hindlimb are displayed. Both fore and hind limbs each have two power law equations; one for symmetrical gaits and another for asymmetrical gaits. This accounts for the sudden drop in duty factor values as the animal transitions from a symmetrical to an asymmetrical gait.

It can also be seen from this figure that as the animal's velocity increases, ground contact decreases introducing at least one suspension phase per gait cycle.

Relative stride length, the ratio of stride length to hip-height, is plotted versus Froude number in the bottom-left of Figure 4.11. It is clear from the plot that as velocity increases, the relative stride length of the animal also increases. To calculate the actual stride length, Equation 4.3 is used. From this value we can then calculate the stride frequency (strides per second) using Equation 4.4.

$$\text{stride length} = \text{relative stride length} \times \text{hip-height} \tag{4.3}$$

$$\text{stride frequency} = \text{velocity} \,/\, \text{stride length} \tag{4.4}$$

The fourth image in the bottom-right of Figure 4.11 displays a plot of the stride frequency. While stride frequency increases with velocity, the rate of increase is not great. As an animal increases its velocity, it does not simply increase the rate at which it moves its limbs. The animal can increase its velocity by extending its limbs farther, achieving greater reach at each step. This allows the animal to cover a greater distance per stride without increasing its strides per second.

4.5.2 Discussion and applications

Robert McNeill Alexander has written multiple books discussing various aspects of animal locomotion and dynamic similarity [6, 8, 4]. In addition to these, dynamic similarity is discussed specifically in relation to

the elastic mechanisms involved in animal locomotion [5] and to cost of transport [7].

The predictions of dynamic similarity are tested through the experimental study of humans travelling in reduced gravity [107, 49, 125]. In terms of gait analysis, the dimensionless nature of the Froude number is found to be useful when scaling data between animals of different body size [156].

In related work that directly pertains to computer animation, dynamic similarity predictions are often used to calculate and verify gait characteristics of a locomotion generation system. These predictions are used to develop and verify horse gait systems [89]. The efficiency and accuracy of robot gaits generated using evolutionary algorithms are examined using dynamic similarity predictions [100].

The uses of dynamic similarity predictions for the experiments presented in this thesis are discussed in the following summary section of this chapter.

4.6 Chapter summary

This chapter begins with a discussion of how the modern horse has evolved through natural and artificial selection. The horse's musculoskeletal system is then described in detail before issues of equine gait patterns, motion and dynamic similarity are addressed.

The information presented in this chapter is not simply prerequisite reading for the experiments presented in later chapters. Many of the sections and corresponding appendices contain knowledge that can be

directly and indirectly used for the creation of realistic horse models and animations.

Information pertaining to a horse's anatomy, specifically the musculoskeletal system, is crucial for the creation of a realistic animal model. Detailed bone measurements and information on joint mobility is necessary for the construction of a rigid body model; specific details are provided in Chapter 5.

Observations of evolution, breeding, conformation and allometry can be used to augment the realism of a scene. A herd situation for example, may include a multitude of animals of differing age, sex and breed. This variance in morphology and motion can be modelled using the biological observations presented in this chapter, as will be seen in Chapter 9.

Besides using biological data for construction of a model, the gait patterns and variations described in Section 4.4.2 and Appendix Section A.4 are used directly in the animation systems presented later in this thesis.

Furthermore, the dynamic similarity predictions discussed in the preceding sections are employed in a system that accurately animates a horse using correct gaits and appropriate transitions, described in Chapter 10.

Dynamic similarity theory is also the key component of the evolutionary algorithm's fitness function used in each of the experiments presented in Part III. Details of this fitness function are given in Chapter 8.

4.7 Part I summary

The objective of Part I was to provide all of the prerequisite and related work for this thesis and in each chapter, decisions regarding direction and experimental approach were justified.

In the first chapter, animation techniques were discussed and some seminal papers on the topics of quadrupedal animation and gait generation were described. At the end of the animation chapter, it was concluded that EC techniques would be explored for the creation of both kinematic and physics-based quadrupedal animations.

In the subsequent chapter on natural computing, biologically inspired algorithms and the field of evolutionary computation were introduced. Three popular evolutionary algorithms were described in detail and contrasted with one another. A grammar-based specialisation of Genetic Programming called Grammatical Evolution was judged to be the most applicable evolutionary algorithm for the purposes of this thesis.

In the final chapter of Part I, biological research was presented. Information on bones, muscles and joints was provided for use in the construction of an animal model. The information on gaits and transitions can be used to reproduce realistic motion in animal animation systems.

In Part III, a series of experiments that employ a Grammatical Evolution-based approach to generating and optimising gait data for various models and animation systems is presented. In each case, dynamic similarity predictions are utilised in the fitness function. Gait pattern data and other elements of animal locomotion are also incorporated in the presented animation systems.

In advance of this, the next part of this thesis describes the construction of kinematic and physics-based horse models. The origins and representations of the gait data are then discussed and attempts to manually generate and optimise that gait data are subsequently presented.

Part II

Model creation and manual gait generation

In Part II, the construction of the horse models and the use of gait and motion data to animate them is described. Animation systems which facilitate the manual generation of motion data are also presented.

In the first chapter of this part, Chapter 5, the data from which a horse model is created is discussed with regard to its origins and skeletal simplifications. The construction and animation of both a kinematic and physics-based horse model is then described.

The gait and motion data used to animate these models is then outlined in Chapter 6. Potential sources of this data are explored and the manner in which the data is represented for the experiments presented later in this thesis is discussed.

In the final chapter of this part, two separate manual gait data development systems are presented; one for kinematic and another for physics-based animations.

Part II concludes with a discussion of how manual motion generation is nontrivial and automated approaches are desirable, such as those explored in the remainder of the thesis.

Chapter 5

Horse model

As mentioned in Chapter 4, the horse has historically played a significant role in human society, for both work and pleasure. To this day, the horse is still prevalent in many aspects of human life. As such, videos and animations of the horse in motion are abundant in film, advertisements and computer games. There is also much information available concerning the horse's physiology and behaviour, in comparison to other animals. Taking these factors into consideration, the exploration of quadrupedal animation approaches presented in this thesis is based upon the horse.

In Chapter 2, various methods for producing quadrupedal animations were described. It was concluded that both kinematic and physics-based models were applicable in different situations. It was also decided that an evolutionary algorithm, specifically Grammatical Evolution, could be used to generate and optimise motion data for animating these models.

In this chapter, details are provided on how both kinematic and physics-based models of a horse are constructed. The construction and animation of the kinematic model, described in Section 5.2, is relatively

simple in comparison to that of the physics-based model. In Section 5.3 full details pertaining to the construction and animation of an articulated rigid body model of a horse are presented. The creation of motion in the model using spring forces is also explained. Before the model construction process is described however, the data from which they are constructed is discussed.

5.1 Data and animation type

The quality and realism of an animal animation is dependent on how precisely the model's real-life animal equivalent is reproduced. Accurately modelling an animal for a computer animation requires some level of knowledge of that animal's structure, however, the necessary level of detail depends on the type of animation that is being produced.

When creating a very basic kinematic animation for example, an artist might refer to an image of the animal. The image may only show the animal's external shape but this could be sufficient to create a 3D polygon mesh approximation. Motion could be produced through deformations of the mesh, however, it could be difficult to create a realistic animation in this fashion.

A more pragmatic approach to animal animation is the hierarchical kinematic model described in Section 2.2.1. Knowledge of an animal's musculoskeletal system can be used to create an articulated figure and biomechanical motion data could be used to animate it; the creation of this hierarchical model is of great importance.

The benefits of using a hierarchical model for kinematic animation were explained in Section 2.2.1. For physics-based animations, the connectivity of a model is established during the construction process and enforced by the physics engine throughout the simulation. Nevertheless, the hierarchical model approach is still beneficial when positioning a physics-based model's rigid bodies, as will be discussed in Section 5.3.1.

Before the construction of these models is described however, the origins and simplifications of the model data are discussed.

5.1.1 Origins of data

There are many different sources of data that can be used for the construction of an animal model.

Dimensional information can be physically measured from a subject, if one has access to an obedient animal. This hands-on approach to data collection is probably unsuitable for most situations.

A more practical source of basic data is an image. The external proportions of an animal can be measured from an image and if multiple images of the same animal are available, a 3D estimation of the animal's morphology can be made. If there is no point of reference of known size in the image however, the actual dimensions of the body cannot be determined.

The motion capture techniques introduced in Section 2.2 can be used to gather dimensional data from an animal. The motion capture system identifies the absolute position of the markers attached to an animal's body and the dimensions and proportions of that animal can be estimated.

The positions of the markers correspond to points on the external surface of the animal but these values can still be used for animation purposes. It may also be possible to estimate an animal's skeletal structure from this data [101].

Although the motion capture approach is attractive, in many cases it is prohibitively expensive and fraught with difficulties when non-human animals are the subject.

Data pertaining to animal anatomy can also be found in publications from the veterinary and biomechanics fields, amongst others. The data may originate from non-invasive studies of live animals [163] or through the dissection of euthanised animals [33].

Source of skeletal data

For the horse models described in this thesis, the skeletal data comes from Dutch Warmblood horses. This breed is chosen because data is available and due to its popularity as a riding horse.

Mass data for the individual body segments of a horse is taken from [33]. This paper provides the mass, centre of mass, density and inertial tensor for 26 body segments of a horse. The presented data was physically measured from six dissected Dutch Warmblood horses and an average of those results is presented. Body segment dimensional data, such as bone length, is also provided.

The above paper does not provide detailed positioning information for the bones in the skeleton. Instead, the joint-angles of an average breed of horse at stance are measured from detailed illustrations [77]. Information regarding ideal conformation, presented in Appendix Section

A.2, is also used to orient the bones and position the neck during the model construction process.

5.1.2 Skeletal simplifications for animation

For animation purposes, certain aspects of the musculoskeletal system can be simplified without compromising on the quality of the resultant animation.

The most important bones in terms of movement and appearance are the long and flat bones. The long bones of the legs rotate in the familiar patterns of natural locomotion and must be modelled appropriately.

Flat bones such as the scapula and pelvis are also very important in the locomotion process. Other flat bones such as the skull affect the balance of a physics-based model as well as being structurally important.

Whilst the shock absorbing behaviour of the short bones is biomechanically significant, they are visibly static. In a very sophisticated physics-based animal model, the short bone physiology could be reproduced but for the models presented in this thesis, they are not directly modelled.

Similarly, as the joints between the vertebrae in the spine are almost static, detailed modelling of the horse's back is unnecessary.

Practically speaking, the skeleton can be simplified based on the joints that are to be modelled. For the horse models presented in this thesis, only the synovial diarthrosis joints are considered. These are the joints with a large range of movement such as the hip and the knee, as described in Section 4.2.2.

Essentially, any set of bones connected by either synarthrosis (little or no movement) or amphiarthrosis (very small movement) joints are treated as a single bone segment.

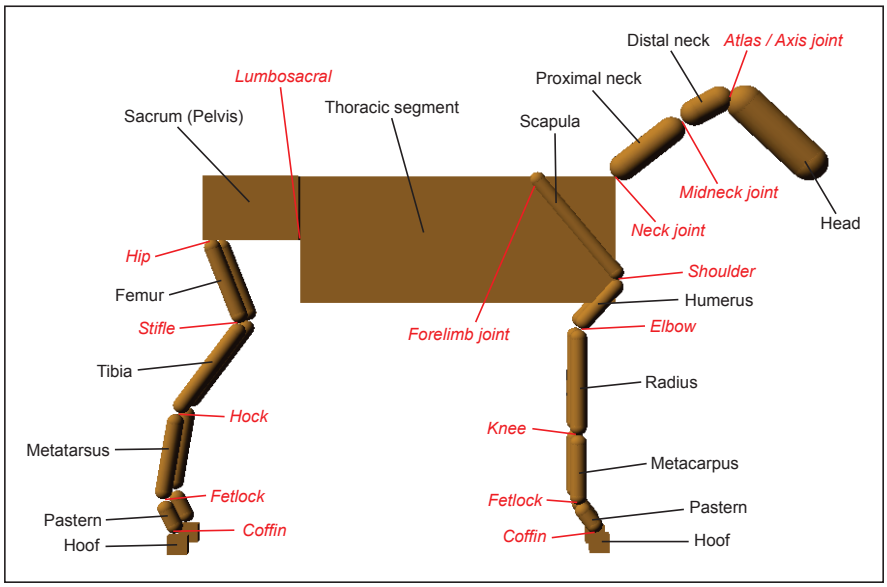


Figure 5.1: Simplified articulated horse model annotated with bone, segment and joint names. Bone and segment names are shown in black and joint names are shown in italicised red.

In Figure 5.1 the simplified articulated horse model is displayed and annotated with names of its bones, body segments and movable joints. The following list gives details of those bones that have been simplified to a single segment and any joint adaptations.

- **Head**: modelled as a single bone segment
- **Atlas and Axis joints**: single joint between head and distal neck
- **Distal neck segment**: includes the 1^{st} - 4^{th} cervical vertebrae
- **Proximal neck segment**: includes the 5^{th} - 7^{th} cervical vertebrae
- **Thoracic segment**: includes 18 thoracic and 6 lumbar vertebrae
- **Forelimb joint**: unnatural joint replaces thoracic sling

- **Radius**: combination of radius and ulna
- **Metacarpus**: combination of cannon bone and splint bones
- **Pastern (fore)**: combination of first and second pastern bones
- **Coffin joint (fore)**: immovable
- **Lumbosacral joint**: between thoracic segment and sacrum
- **Sacrum (Pelvis)**: includes 5 sacral vertebrae
- **Tibia**: combination of tibia and fibula
- **Metatarsus**: combination of cannon bone and splint bones
- **Pastern (hind)**: combination of first and second pastern bones
- **Coffin joint (hind)**: immovable
- **Tail**: omitted

The degrees of freedom of the joints in the simplified model are presented in Table 5.1. Hinge joints have rotation along the sagittal plane about a single axis. Those joints marked "ball and socket" are actually modelled as a slightly more constrained joint for simplicity; the joint is capable of flexion, extension, abduction and adduction but no rotation.

The actually values used in the construction of the horse models are presented in Appendix Section B.1.1. The values provided relate to bone dimensions, angulation and mass. The bone segment mass values are only required for physics-based model construction, which is described later in Section 5.3. Prior to this, the construction of a simpler kinematic horse model is described.

Table 5.1: Equine joints and degrees of freedom (ball and socket joint does not allow rotation).

Joint Name	Connects from	Connects to	Type
Atlas/Axis	Head	Distal neck	ball and socket
Midneck	Distal neck	Proximal neck	ball and socket
Neck	Proximal neck	Thoracic	ball and socket
Forelimb	Thoracic	Scapula	ball and socket
Shoulder	Scapula	Humerus	ball and socket
Elbow	Humerus	Radius	hinge
Knee	Radius	Metacarpus	hinge
Fetlock (fore)	Metacarpus	Pastern	hinge
Coffin (fore)	Pastern	Hoof	static
Lumbosacral	Thoracic	Sacrum	hinge
Hip	Sacrum	Femur	ball and socket
Stifle	Femur	Tibia	hinge
Hock	Tibia	Metatarsus	hinge
Fetlock (hind)	Metatarsus	Pastern	hinge
Coffin (hind)	Pastern	Hoof	static

5.2 Kinematic model

As described in Section 2.2, a kinematic model is animated without consideration of the physical forces that cause movement.

The kinematic horse model presented in this section is represented as a hierarchical kinematic model (see Section 2.2.1). Using the data presented in Appendix Section B.1.1, the model is created and animated using OpenGL.

5.2.1 OpenGL

OpenGL (Open Graphics Library) is a graphics system that facilitates the creation of interactive programs to produce colour images of moving three-dimensional objects [176].

OpenGL is a software interface to graphics hardware. It provides commands that can be used to specify the objects and operations involved in the creation of these programs.

Complicated models can be constructed from OpenGL's small set of simple geometric primitives which comprise points, lines and polygons. Other aspects of rendering such as colour, texture and lighting can also be specified.

The objects of an OpenGL scene are arranged in 3D space and a camera position from which the scene is viewed can be defined. Once the mathematical description of a scene is complete, it is converted to pixels on the screen through a process known as rasterisation.

OpenGL offers three transformation routines for positioning the objects in a scene; translate, rotate and scale. By applying combinations of these transformations, objects can be positioned relative to a global origin or each other.

For hierarchical modelling, the OpenGL matrix stacks can be manipulated so that the transformation routines applied to a particular node in a tree structure will affect other nodes lower in the tree, thus enforcing the connectivity constraints discussed in Section 2.2.1.

A more detailed description of OpenGL is beyond the scope of this thesis, however, the full OpenGL specification is available [95].

In the following section, the construction of a hierarchical kinematic horse model using OpenGL is described.

5.2.2 Model construction

The kinematic horse model is constructed using the bone and body segment data provided in Appendix Section B.1.1.

The data is first put into a quadruped model description file. A template of this bespoke file format is shown in Appendix Section B.1.2 and an example of the actual file used for many of the horse model experiments in this thesis is presented in Appendix Section B.1.3.

This data file is then read in by the animation system. The structure of the file format pertains specifically to a cursorial quadruped animal and the animation system anticipates data for specific body segments; namely a trunk, four legs and a necklike appendage.

Each segment can contain any number of bones and a corresponding number of joints. If there are n bones in a segment, the system expects *(n - 1)* joints to be defined. The bone segments each have a type, name, mass, length, radius and angulation value. For square bones, the radius value is replaced by width and height values.

Each joint description includes a name, type, set of joint limit values (see Section 5.3.1) and names of the two bones it is to join. Additional joint definitions are also provided to describe how the segments themselves attach to the trunk segment.

Once the model data is successfully read in, the animation application interprets the data as a hierarchical quadruped model and uses the OpenGL primitives, transformation routines and matrix stack manipulations to construct the quadruped model accordingly.

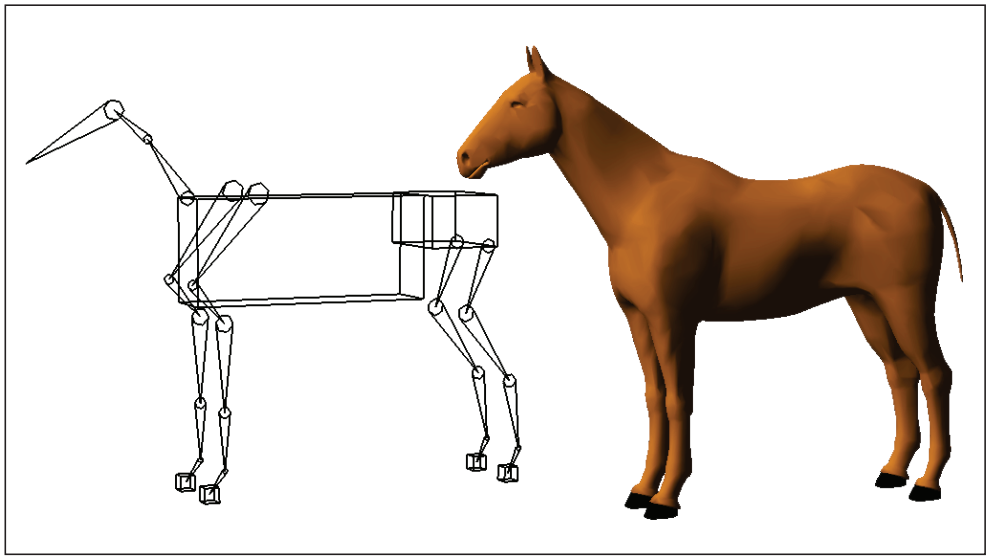


Figure 5.2: Model construction: skeletal (left) and skinned (right).

The model's skeletal structure is displayed in the left image of Figure 5.2. The long bones of the model are represented as a 2D shape comprising two lines which stem from a circle around the upper joint, and taper to a point at the lower joint. Body segments such as the thoracic segment, sacrum and the hooves are modelled as rectangular cuboids.

The basic wireframe model can be skinned with a realistic looking 3D polygon mesh, as shown in the right image of Figure 5.2. As the underlying skeleton moves, the mesh will deform with it.

This static model is straightforward to construct with OpenGL; the hierarchical model is read in from the file and the appropriate transformations produce a model in the anatomical position. When a model is constructed in this hierarchical manner, creating an animation is also relatively simple.

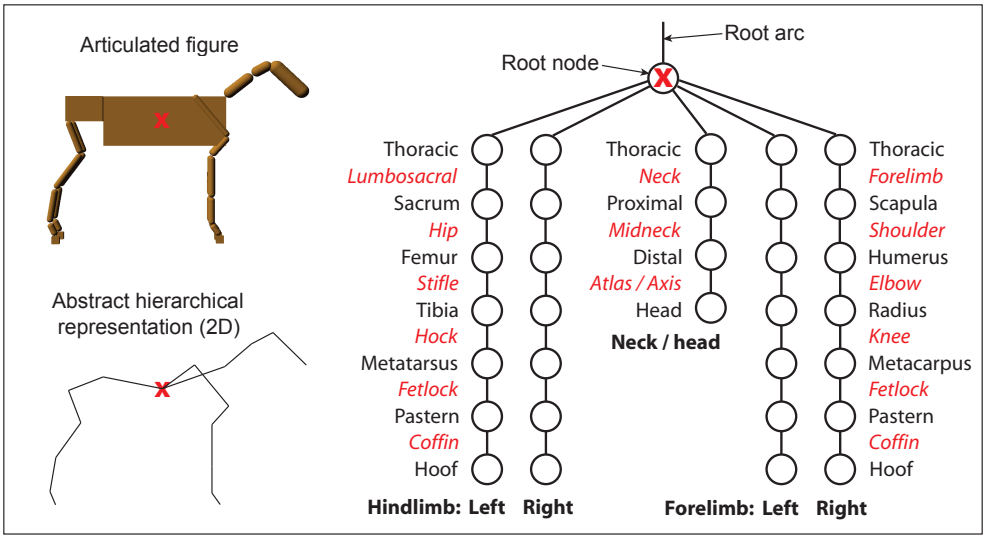


Figure 5.3: Illustration of how the articulated horse model is represented as a tree structure. Bone and segment names are shown in black and joint names are shown in italicised red.

Creating motion in the model

The hierarchical horse model is displayed in Figure 5.3. Motion in a limb is produced by applying rotation transformations to the nodes (bones) in that limb's branch of the tree structure.

Starting from the uppermost node in a limb's branch, the rotation of each bone in that limb's hierarchy propagates downwards through all the other bones. The model is animated by applying a specific pattern of rotations to each limb's bones, for every frame. The entire model is also translated a calculated distance, horizontal to the ground plane to indicate locomotion.

For a realistic animation, the bones must be rotated according to some motion data which will be discussed in Chapter 6.

In the following section, the construction of the physics-based horse model is described.

5.3 Physics-based model

In contrast to the kinematic model described in the previous section, the motion of a physics-based model takes mass and force into consideration.

The physics-based horse model, used in the experiments presented in this thesis, is constructed as an articulated rigid body model, as described in Section 2.3. The physics engine used is ODE (Section 2.3.1) and the graphical output is created using OpenGL (Section 5.2.1).

The data used to create the model is the same as that used for the kinematic model and the file format is identical.

In the following section, the construction of the physics-based model is described in terms of ODE geometric primitives, positioning and connectivity.

5.3.1 Model construction

As with the kinematic animation system, the horse model data file, provided in Appendix Section B.1.3, is read in by the physics-based animation system. The various bone and segment rigid body objects are created, positioned and connected together by joints, as described in the following subsections.

Bones and segments

The long bones of the model are represented using the ODE capsule geometry. A capsule has adjustable mass, length and radius. The radius variable determines the radius of the cylinder and the hemispheres which cap each end.

Capsules are a suitable choice for the long bones for several reasons. They resemble the general shape of a long bone but more importantly, collision detection between capsules and other objects is very efficient due to their shape. The rounded ends also allow for unimpeded rotation.

Two capsules connected end-to-end can rotate about all axes as long as a sufficient buffer space is present. This space prevents the capsules from colliding with each other during rotation. Inserting a space in this manner is essentially emulating the naturally occurring lubricating liquid and shock absorbing material found between bones.

For the hooves and trunk of the model, capsule objects are not suitable. In the case of the hooves, the capsule object would not provide the flat surface necessary for interactions with the ground plane. Similarly for the trunk, a capsule does not provide the appropriate connective surfaces, therefore rectangular block objects are used instead. The blocks are rectangular cuboids with adjustable mass, length, width and depth.

The ODE capsule and block objects are graphically displayed using OpenGL. The capsules are modelled using the OpenGL Utility Library's cylinder shape, capped with hemispheres at each end. The blocks are drawn using simple polygons.

Segment positioning

Once the properties of the model's rigid bodies are defined and created, they can be positioned. The model data file contains no information about absolute positioning of the bones and segments. Instead, the animation system calculates the positions of the rigid bodies in a pseudo-hierarchical manner.

As positional information is initially not available, the vertical positioning of the trunk is unknown until the vertical height of the limbs is calculated. As such, the legs, trunk and neckhead segments are treated as independent hierarchies and constructed separately in some arbitrary initial position.

Once constructed, the trunk can be vertically positioned based on the limb heights, and the limbs can be horizontally positioned based on the dimensions of the trunk. With these segments in place, the neckhead segment can be positioned relative to the rest of the model.

Bone positioning

Within the individual segment hierarchies, the position of a bone is determined by the position of the point it attaches to on a previously placed bone.

When specifying the position of a bone, the positional values are supplied to ODE in global coordinates and relate to the centre point of that bone. The placement of every bone in the hierarchy must be calculated using the absolute position of the previously placed bone, compensating for the orientation of both bones.

The hemispherical shape of the capsule's ends must also be taken into account. When a capsule is rotated for example, the attachment point may no longer be prominently accessible for connection, i.e. some of the rest of the capsule's hemisphere may obstruct a direct point-to-point connection. The position of the joining capsule must be offset to compensate for the impediment of the capsules' hemispheres.

Once the position of each rigid body in the hierarchy is calculated, the bodies are connected together using ODE's supplied joints, according to the list of joints shown in Table 5.1 of Section 5.1.2.

This description of the model construction process is simplified but comprehensive details of the construction process are given in Appendix Section B.1.4.

Stance and joint limits

When the physics-based horse model is constructed and the simulation loop begins, the articulated rigid body model simply collapses under the force of gravity, like the rag doll horse model described in Section 2.3.

To counteract this, at stance, the joints in the model are locked at their stance angle using ODE joint limits. The joint limits are parameters of every joint in ODE. They specify the maximum amount of rotation allowed about each active axis in the joint and a joint can be completely locked by setting all of the joint limits to zero.

Joint locking prevents the model from collapsing under the force of gravity, without the application of any torques on the joints. This locking action emulates a horse's ability to conserve energy by physically locking its joints when at stance.

These locks should not be confused with the natural joint constraints and limits of the horse's body, which describe the axes and angles through which a joint can rotate without physical impediment.

These natural limits can be included as a rotation constraint in the horse model, ensuring that no bone will move into a position that would be unnatural or impossible for a horse to achieve.

When movement is required, the locks are released by setting each joint's limits to their natural limit values, as specified by the model input data file. At this point the motion controllers are responsible for moving the limbs in a specified pattern.

5.3.2 Motion

To produce movement in the model, torques of specific magnitude and direction are applied by motion controllers about each of the joints in a limb.

A single interlimb motion controller controls the overall timing of each limb's movement, to produce a recognisable gait.

Each limb in the model also has an intralimb motion controller in command of its bone rotations. The intralimb motion controller must rotate the bones according to some pattern, so that the resulting limb movement is aesthetically realistic and helps propel the model at the expected velocity as its hoof pushes off the ground surface.

For realistic movement, both motion controllers are supplied with appropriate motion data.

The gait pattern data which is used by the interlimb motion controller contains limb phase difference values for each of the natural gaits, as described in Section 4.4.2. Using this data, the interlimb motion controller controls the timing of the four intralimb motion controllers.

Joint-angle motion data is supplied to the intralimb motion controllers and determines the rotations applied to each bone. The origin and representation of the joint-angle motion data is discussed in Chapter 6.

During simulation, a single joint's motion data for a single gait cycle is stored as a set of 100 discrete points. Each of these points represents a target joint-angle for a specific time in the gait cycle. The simulation steps through these points at a rate dependent on the current stride frequency value.

At each step, the current angle of a joint is compared to its target joint-angle. In an effort to rotate the attached bone to this target angle, an appropriate torque is calculated and applied to the joint.

Torque calculation

A simple to implement and commonly used method for calculating a torque such as this is the Proportional Derivative (PD) controller [103, 119]. A PD controller outputs a torque proportional to the difference in position and velocity between the actual and desired states of an object and therefore provides a simple, low-level control mechanism [113].

A PD controller-based system is reliant on the trial and error tuning of controller parameters which can be problematic. Alternative methods can involve complicated inverse dynamics calculations or sophisticated feedback systems. The simplicity of the PD controller implementation and behaviour make it an attractive solution for the physics-based models presented in this thesis.

The behaviour of a PD controller is basically that of a spring-damper system and will be described as such in the following section.

5.3.3 Spring-damper system

The torque required to rotate a bone to some target joint-angle is calculated using a spring-damper equation based on Hooke's spring law [72].

Hooke's law of elasticity allows one to replicate springlike behaviour in a computer simulation. In simple terms, Hooke's law states that strain is proportional to stress and is mathematically stated in Equation 5.1.

$$F = -kx \tag{5.1}$$

where F is the calculated Force, k is the spring coefficient and x is the displacement of the end of the spring from its equilibrium position.

The value of k determines the tightness of a spring, with larger values indicating a tighter spring that will stretch less per unit of force; smaller values signify a looser spring. According to Hooke's law, if one were to extend a spring with a weight on the end and release it, the spring would oscillate indefinitely about its equilibrium point. In reality, the spring would eventually come to rest due to a damping force that is proportional to the difference in velocity between the two ends of the spring [154].

A spring-damper system can be modelled using Equation 5.2.

$$F = -kx - bv \tag{5.2}$$

where F, k and x are as above, b is the damping coefficient and v is the relative velocity between the two ends of the spring. The larger the value of b, the quicker the object will come to rest.

The physics-based animation system exploits the fact that Hooke's law can be used to calculate a force, that when applied to a bone, results in a rotation about a joint. Using this property for limb motion, given the current angle of a joint, the force, or torque in this case, required to rotate the attached bone to some target angle can be calculated.

The specific spring-damper equation used in the physics-based system is presented in Equation 5.3.

$$T = -kx - bv \tag{5.3}$$

where T is the calculated torque, k is the spring coefficient, x is the current displacement from the joint's target joint-angle, b is the damping coefficient and v is the current angular velocity of the attached bone.

The greater the distance between the current joint-angle and the target angle, the greater the magnitude of torque required to move the bone to its target position.

Provided that the spring and damper coefficients are set appropriately, the motion controllers can keep the bones rotating towards the angles suggested by the target data, at every time step.

This spring-damper approach can produce smooth, natural looking motion, however, there is one major drawback. Each joint in the model must have its own set of appropriately set spring and damping coefficients and calculating these values is nontrivial.

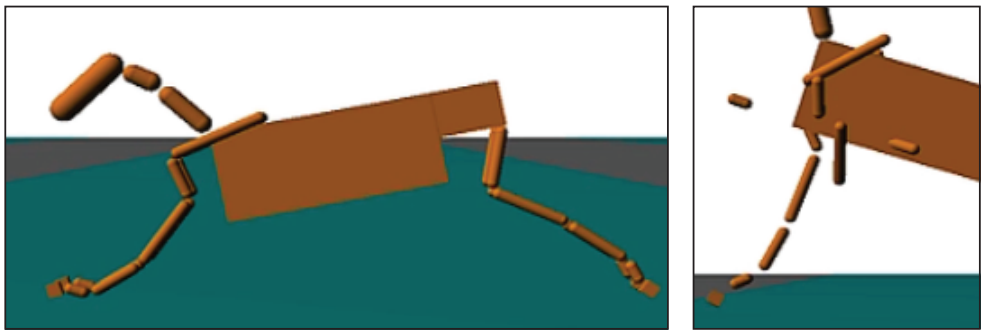


Figure 5.4: The effect of spring-damper coefficients with values set too low (left) and set too high (right).

Simulation instability

In Figure 5.4, the effect of incorrectly set spring coefficients in the physics-based horse model is illustrated. The image on the left shows the horse model being simulated with some of its joints' spring coefficients set to too low a value.

In this example, the target joint-angles are constantly set to the model's stance angles and the joint locks are removed. The torque values calculated by the spring equation are insufficient to hold the body up against gravity and the model sinks to the ground.

Video 5.1 Spring values set too low

Video 5.2 Spring values set slightly too low or too high

Video 5.3 Spring values set correctly

Video 5.4 Spring values set too high

Video 5.5 Spring values set too high (slow motion)

In the right image of Figure 5.4, the simulation has become unstable and has exploded. In this case, a spring coefficient is set to too high a value

in at least one joint. At some point in the simulation, the displacement of a joint from its target angle may have become quite significant, and the torque calculated by the spring-damper equation was correspondingly large.

When the torque was applied about the joint, it may have caused the joint to overshoot its target. At the next timestep, a restorative torque, in the opposite direction, may have been applied. This dynamic continues for a number of time steps, exacerbating the problem, until the torques being calculated are so large, the joints of the model can no longer hold together and the simulation explodes.

Ideally the spring and damping coefficients will be set for each joint so the attached bones smoothly rotate to within a very small distance of their target angle at each timestep. The exact values necessary to achieve this can depend on several factors.

The position of a joint in the limb hierarchy has significant influence on the spring and damping coefficients. As each bone in a limb has its own mass and angular velocity, as a joint attempts to rotate its attached bone, it may be doing so with or against the momentum of the entire hierarchy at that point in time.

Other contributing factors include ground reaction forces and the degree to which the joint has to flex and extend depending on the motion.

In Chapter 7, the manual setting of these spring and damping coefficients is described using a manual physics-based motion generation environment. Automated approaches to coefficient generation are also discussed in Chapter 9.

5.4 Chapter summary

The chapter begins with a discussion of the sources of data that can be used in the construction of horse models for computer animation.

One specific source of data is described and details are given as to how that data is simplified for use in model construction. Using this data, the construction of a hierarchical kinematic horse model is described and OpenGL is introduced.

Following on from this, the construction of a physics-based model is described in detail and its motion generating system using spring-damper equations is presented.

The chapter concludes with mention of the issues arising from setting spring-damper coefficients which will be discussed in later chapters.

The models described in this section are used for the experiments described in Part III and also in the manual motion generation environments introduced prior to this in Chapter 7.

In advance of this, the actual motion data that these models use for animation is discussed in the following chapter.

Chapter 6

Gait motion data and representations

Motion data measured from animals can be used to increase the realism of an animation. In this chapter, the data that determines how the individual bones in a model are rotated is discussed. This type of data is referred to as motion data or joint-angle data. This is distinct from gait pattern data which simply states the phase difference between the limbs.

The motion data can be anything from a simple set of visual observations of animal movement to complex measurements of skeletal motion. Biomechanical knowledge can also be used to calculate how certain segments of an animal's musculoskeletal system behave. Once a set of data is compiled, it must be represented in a manner that the animation system can interpret. In general, data representations should be convenient to use, compact and ideally intuitive.

In the later sections of this chapter, several motion data representations are presented. Firstly however, the origins of the motion data used for the experiments in this thesis are discussed.

6.1 Data origins

In an ideal situation, an abstract model exists that allows for the calculation of the individual bone motions necessary to produce movement, for any gait and for any given skeleton. Unfortunately, no such model exists and therefore some form of motion data must be acquired and standardised to a format and representation that an animation system can use.

As realism is imperative, animation approaches that rely on human interpretations of animal movement are disregarded. Instead, the focus is on taking measured motion data from a reliable source and providing it to the animation system's motion controllers in an appropriate format.

This motion data can be obtained from various sources including photographs, video, motion capture and data published in the biology and veterinary literature. In the following sections, each of these data sources is discussed.

6.1.1 Photographic

Often a still image of an animal in motion can yield more information than real-life visual observation or a video. A single photograph of an animal in motion literally gives a snapshot of an animal's physical state at a point in time. Whilst some information can be taken from an individual

image, a fuller view of animal motion is found within the context of the frames that precede and follow it.

The study of animal motion through high-speed photography was pioneered by Eadweard Muybridge and Étienne-Jules Marey in the late 1800s. Marey, a French scientist, studied the motion of animals initially by way of physical instrumentation and subsequently through photography [118]; in 1882 he developed the field of chronophotography. Using a bespoke chronophotographic gun, Marey was able to capture twelve consecutive frames of an animal's motion on a single image. These composite images were then used to study the motion of many different animals.

A decade prior to this, in 1872, the question of whether a horse at gallop experienced a suspension phase was a hotly debated one; Marey himself had written at the time that a galloping horse does experience a brief moment in which all hooves are off the ground simultaneously. The quest for a definitive answer prompted the 19th century English photographer Eadweard Muybridge to produce high-speed photographs of animal locomotion [38].

To capture the motion of an animal, a series of cameras were positioned side-by-side along a runway. Thin silk tripwires were then stretched across it and attached to triggers on the cameras. An animal subject was coaxed into running along the runway and as the animal progressed, its legs tripped the silk wires, causing the cameras to take a picture at the same instant.

In the photographic example shown in Figure 6.1, it can be clearly seen in the second frame that a galloping horse does indeed experience a period of suspension [136], finally answering a longstanding question.

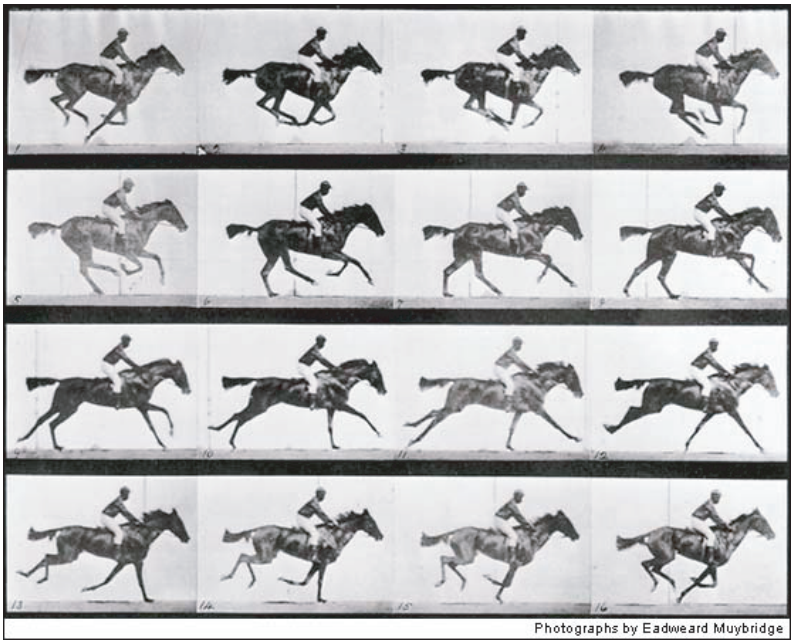


Figure 6.1: A Muybridge photograph of a galloping horse [136].

The high quality and extensive collection of Marey and Muybridge images are still considered to be an invaluable resource to biology and veterinary researchers, as well as to artists and animators.

For this thesis, the Muybridge photographs were considered as a potential source of motion data, however, it was experimentally determined that manually extracting data from images is time-consuming and inaccurate as the simple kinematic animation listed below illustrates.

Video 6.1 Animation from Muybridge photographs

An automated approach for extracting motion from raw videos is the subject of ongoing research, however, this continues to prove a very difficult task [155]. Consequently, motion capture techniques persist as a popular source of motion data.

6.1.2 Motion capture

Full 3D motion capture can yield highly accurate motion data measured directly from a subject. The motion capture of an object involves the sensing, digitising and recording of that object as it moves. The animations produced from this data are often highly realistic [123].

The motion capture subject is usually instrumented in some way so that the key features on the body can be detected and recorded. These key feature positions are matched to corresponding points on a computer constructed model which then moves according to the recorded motion. Although there is a large amount of human motion capture data freely available or procurable at low cost, the same cannot be said for animals.

Before the motion capture process can begin, an animal must be obtained that will produce a desired motion on demand, or can be trained to do so. When capturing gait pattern motion data, large animals such as horses are often trained to run on a treadmill. The animal's performance can be unpredictable however, and the capture process can be fraught with difficulty. Figure 6.2 shows several still images from a motion capture driven horse animation. This animation of a bucking horse is an example of an animal misbehaving during a motion capture process.

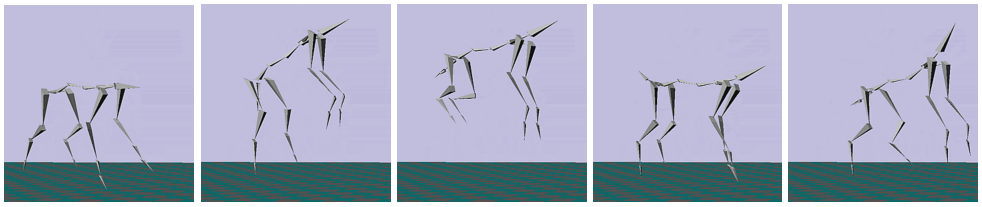


Figure 6.2: Motion capture of a horse. The sequence of images shows a horse bucking intensely.
Video 6.2 Motion capture horse model

Some companies specialise in the production and sale of animal motion capture data, but it is often prohibitively expensive. For the vast majority of animals, motion capture data is simply unavailable and unattainable.

Other less expensive 2D motion capture techniques are also possible but their practicality and quality of results are unknown.

A 3D real-time physics-based model of a trotting horse, whose motion data is extracted from a 2D video motion capture of a horse using an active contour technique, is described [119]. The quality and expense of this approach are undisclosed and it is stated that there are problems with interlimb occlusion due to the 2D nature of the motion capture.

A cine-radiographical system is used to capture the gait cycles of a hedgehog [200, 138]. Using an x-ray machine, a glass screen, a brightness amplifier and camera, the locomotion of a hedgehog on a treadmill is captured. Again, the quality of the captured motion is unknown and this x-ray machine approach is probably unsuitable for larger animals.

Overall, a motion capture approach to animation can produce highly realistic results if it is possible to capture the subject's motion. The cost of motion capture however, is prohibitive in many cases.

In the following section, the use of the data that is found in research publications is discussed.

6.1.3 Published

In the past century, animal gaits have been studied and analysed in considerable detail. Consequently, significant amounts of motion data are published in the biological and veterinary literature.

This data can be extracted and used for the animation of animal models. Unfortunately, the data may be non-uniform, noisy and it is often measured from multiple animals of different breed, conformation, size or sex.

Any extracted data must undergo a process of formatting and standardising. Once the acquired data is in a suitable format however, it may still not be immediately usable for animation.

It is often the case that morphological information is unavailable for the animal from which the data was measured. The extracted data must therefore be adjusted to "fit" a particular computer constructed model.

Manual tuning of this data is an inaccurate and tedious process of trial and error, as will be seen in Chapter 7. It is possible however, to use the untuned, raw data as the basis for automatic motion generation.

This automatic process will be discussed in Part III as all of the presented experiments are concerned with automatic motion generation.

Data source

The motion data used as the basis for all experiments in this thesis is taken from published joint-angle data plots which relate to the forelimb [17] and hindlimb [18] of a trotting horse.

The actual plots and details of the motion data are provided in Appendix Sections B.2.2, B.2.3, B.2.4 and B.2.5.

Each plot shows a specific joint's angular motion for the duration of a gait cycle. Data values are extracted from each of the plots using a software application [29] which allows for the sampling of plotted data at regular intervals along the x-axis. This set of extracted values is the basis

for all of the manual and automatic gait data experiments presented in this thesis.

The unoptimised motion data does not provide a stable gait cycle with which the physics-based horse model described in Chapter 5 can locomote. This can be cleary seen in the animation listed below.

Video 6.3 Published horse motion data (unoptimised)

Before the data can be used by the animation system (such as in the above video) the data values must be converted from their raw form. In the following section, several different representations for this data are discussed.

6.2 Representations

The motion of a bone can be described by a sequence of discrete values, each of which representing the bone's orientation at a point in time.

Motion data stored in this manner can be directly used by an animation system which steps through the data and rotates a model's bones accordingly.

This idea is illustrated in Figure 6.3. The set of numerical values shown represent the angular motion of a single bone, sampled at regular intervals. The corresponding motion curve is plotted beneath.

The articulated bodies shown in this figure demonstrate how the discrete values can specify a bone's rotation. If this sequence of rotating bone images are viewed in quick succession an animation is produced.

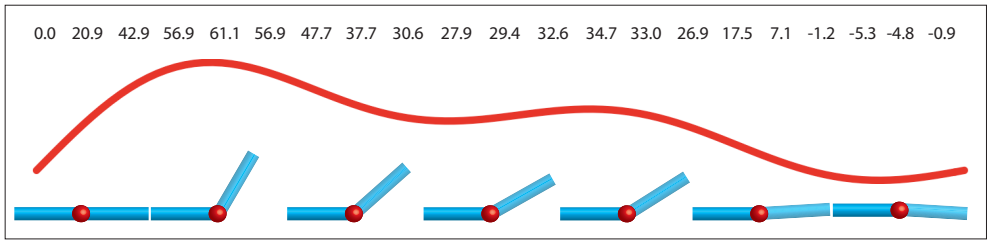


Figure 6.3: A sequence of discrete values are shown as a curve. The bone angles represented by this motion curve are illustrated by the articulated bodies shown in the bottom of the figure.

This approach to motion data representation is perfectly valid and simple to produce. The motion data discussed in the previous section, for example, consists of a small set of numerical values which describe a curve.

By interpolating between these values, a bone angle value can be calculated for each animation timestep and the model can be moved accordingly.

The sample data shown in Figure 6.3 describes a single motion sequence, for a single bone in a specific model. If one wishes to modify and reuse this data however, the numerical representation can prove problematic.

To change the shape of a motion curve, multiple values in the set must be changed in a manner that preserves the natural smoothness of the curve whilst producing the desired motion. This is a nontrivial task and a more intuitive data representation is desirable for this reason alone.

It would be possible to avoid this data modification problem entirely by storing a large library of motions in this numerical representation but as mentioned previously, large sets of data pertaining to a single animal are unavailable.

The solution to this lack of motion data, and a major topic of this thesis, is the development of motion generation approaches. In the later chapters, we present several methods for taking a single piece of motion data and tuning it for use with multiple models, therefore using it as a template for generating numerous different motions.

This reuse process is based upon the optimisation of motion data. When motion data is stored as a set of simple numerical values, each value is an optimisation parameter. This results in a huge search space which includes a great number of invalid solutions, largely due to the smoothness constraint.

For this reason, it is desirable to describe each bone's motion by a smaller set of parameters, which minimises unnecessary information and reduces the search space.

In the following sections, two motion data representations are presented. In Section 6.2.2, a representation in which a curve is described by its distinguishing features is detailed. Before this, a summation of sinusoids representation is discussed.

6.2.1 Sinusoids for cyclical motion

A bone's motion data can be represented as a summation of sinusoidal functions of differing frequencies and amplitudes.

The rotation of a bone about a joint is often the product of multiple muscles pulling on it and these muscles tend to relax and contract in a sinusoidal manner. The waveforms in a summation of sinusoids motion data representation are analogous to a muscle's contribution to a bone's rotation.

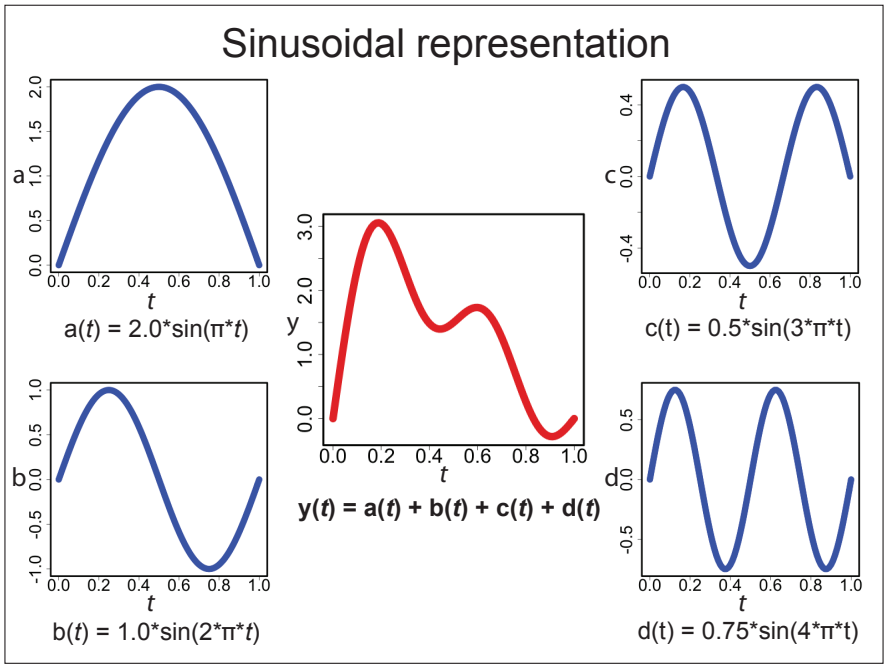


Figure 6.4: Summation of sinusoids motion data representation. The functions a(t), b(t), c(t) and d(t) are added together to give y(t) which contains the rotation data for a single bone for a full gait cycle.

The summation of sinusoids representation is illustrated in Figure 6.4. In this figure, four separate sinusoids of differing frequencies and amplitudes are added together. The resultant waveform can be used to represent a bone's rotation as a function of time.

This representation is more compact, elegant and intuitive than the numerical equivalent discussed in the previous section. It also allows for intuitive adjustment of the motion data through the addition of sinusoids of differing frequencies and amplitudes. By ensuring that each sinusoid added to the summation has the same phase value, the resulting waveform is always cyclical, which is important if a single set of waveforms is to be repeatedly used for sustained locomotion.

Before any data can be modified however, the raw numerical description of a motion curve must first be converted to a summation of sinusoids representation. This is done by Fourier analysis.

Fourier analysis

Fourier analysis is a process by which a signal can be decomposed into separate sinusoidal functions for a range of frequencies [31].

The sinusoidal nature of muscle motion suggests that the motion data naturally lends itself to Fourier analysis as illustrated in Figure 6.5.

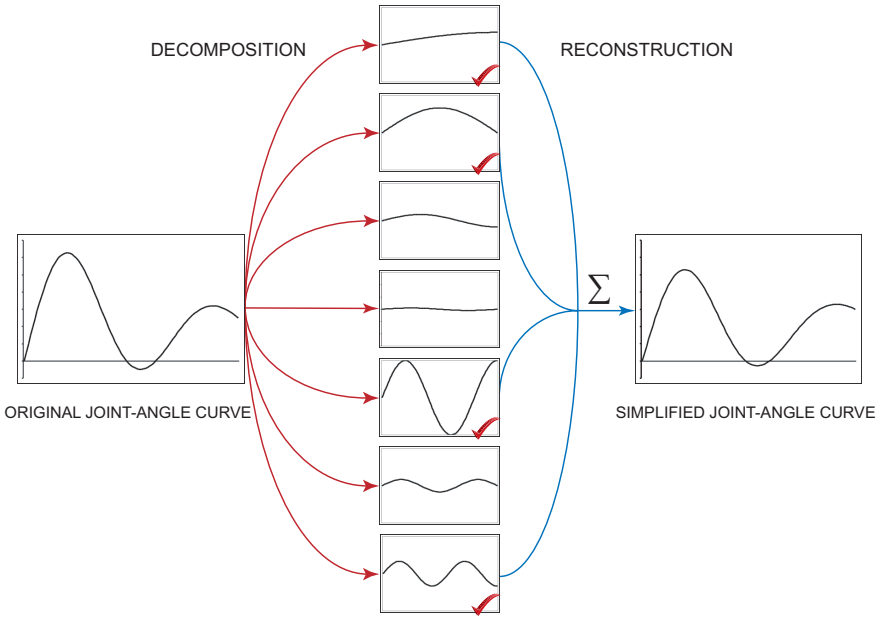


Figure 6.5: Fourier analysis decomposes motion data into a number of sinusoidal functions whose parameters are used to represent the motion data.

A Fourier transform is applied to a motion data curve represented by a sequence of numbers, such as that shown in Figure 6.3. This process yields a set of separate sinusoidal functions of differing frequencies and amplitudes, as is illustrated in Figure 6.5.

Although this set of sinusoids describes the motion data curve, the representation can not be described as minimal as there are still a large number of parameters involved.

To simplify the representation, only the most influential sinusoids are recombined to form the minimal motion data representation. The amplitude of each sinusoid is examined and if it is below some specified threshold value, that sinusoid is discarded, as illustrated in Figure 6.5.

This process may seem arbitrary and lossy, however, this minimal representation is more intuitive. It reduces the optimisation search space and is easily derived from a grammar, as will be discussed in Part III.

The data that will be optimised should be viewed as a template of animal motion. The most influential peaks can be considered as the major muscles involved in locomotion. The smaller details of the curve that are discarded may be the result of less significant muscles, interference from other muscle groups or propagating forces from ground impact.

The summation of sinusoids representation is excellent for storing repetitive cyclical motions such as gait cycle motion data, however, for the acyclical motion data used for gait transitions, described later in Chapter 10, a piecewise representation is used.

6.2.2 Piecewise for transitional motion

The piecewise representation describes a motion curve by dividing it into a series of segment types, each with a position and value.

An example of the piecewise representation is presented in Figure 6.6. In this example, the same sinusoidal curve as shown in Figure 6.4 is represented in the piecewise form.

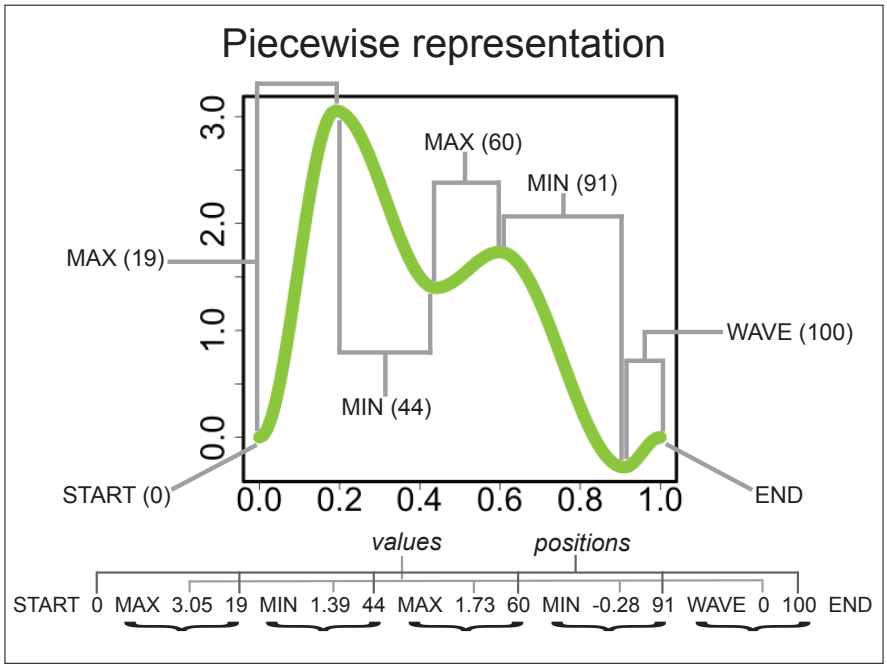


Figure 6.6: A piecewise representation string is presented and plotted with annotations. This string contains rotation data for a single bone for a full gait (transition) cycle.

The piecewise description of a motion curve is displayed at the bottom of the figure. Segment names are shown with their values and positions. The plot of the curve itself is annotated with lines indicating which sections of the curve correspond to each segment.

The majority of the segments are modelled using sections of a sinusoidal waveform calculated using Equation 6.1.

$$s(t) = \frac{sin(t \cdot \pi - \frac{\pi}{2}) + 1}{2} \tag{6.1}$$

where t is a uniformly varying input from 0 to 1. The output s, is also in the range 0 to 1 but rather than changing uniformly, it starts off slowly, then speeds up and then slows down again [155]. The same

shape of curve, but starting at 1 and moving to 0, can by calculated with Equation 6.2.

$$s(t) = \frac{sin(t \cdot \pi + \frac{\pi}{2}) + 1}{2} \tag{6.2}$$

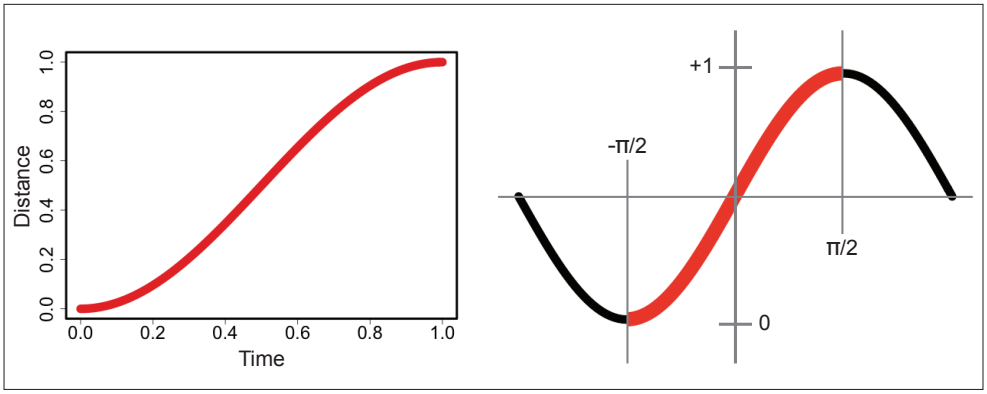


Figure 6.7: The curve segment calculated by Equation 6.1 is shown on the left. This segment is highlighted in relation to a longer sine wave on the right.

In Figure 6.7, the curve produced by Equation 6.1 is plotted in the left image of the figure. The same curve is then shown highlighted as part of a longer sine wave in the image on the right.

In the piecewise representation, a curve is usually totally described using a combination of the segments calculated by Equations 6.1 and 6.2. The following subsection provides a summary of the piecewise segment types and the parameters they each take.

Segment types

In the following list, the various features and parameters of the piecewise representation are described.

In each case, a segment's starting value is the last calculated value of the previous segment. With the exception of PLATEAU, the length of a segment is the difference between the previous segment's ending position and the position value of the current segment.

START *value*:

the curve's initial offset is defined by *value*

MAX *value position*:

curves up to *value* using Equation 6.1

MIN *value position*:

curves down to *value* using Equation 6.2

WAVE *value position*:

curves in either direction to *value* using Equation 6.1 or 6.2

PLATEAU *value start_position end_position*:

maintains a constant *value* between *start_position* and *end_position*

END :

representations end with this feature

For the PLATEAU segment type, a WAVE curve is automatically used to join the previous segment's value to the PLATEAU value at the PLATEAU's *start_position*.

The feature names MIN, MAX and WAVE may appear redundant, however, they serve a purpose.

Assuming all curve segments, with the exception of the PLATEAU, should be sine wave segments, the segment's *value* and *position* param-

eters, along with the previous segment's *value*, should be enough to determine the direction of the segment i.e. curving up or down.

The reason that the feature names are included is to make the representation more human readable and to allow motion constraints to be enforced during optimisation.

Take for example an optimisation process that may only modify the numerical parameters of a piecewise representation's features and those features are "START ... MIN ... ... MAX ... ...",

By including the feature type names, the range that a value can be changed to is limited. The value of MIN for example, is not permitted to be larger than the value of the last segment. Similarly the value of MAX can't be smaller than the value of the last segment.

Use of piecewise representation

Besides performing parameter optimisations on piecewise represented motion data, entirely new motion data can also be created by concatenating multiple feature types with parameters chosen from a range appropriate to that feature's context.

The advantage of the piecewise representation is that it can represent acyclical motion and be easily modified. Figure 6.8 shows an acyclical piecewise data representation connecting two offset cyclical waveforms.

As previously discussed, the motion data of a gait cycle is cyclical, however, different gaits may exhibit different bone motion patterns. The waveforms that represent these patterns may therefore be offset from one another. A waveform that is to join the offset cyclical curves will have to be acyclical.

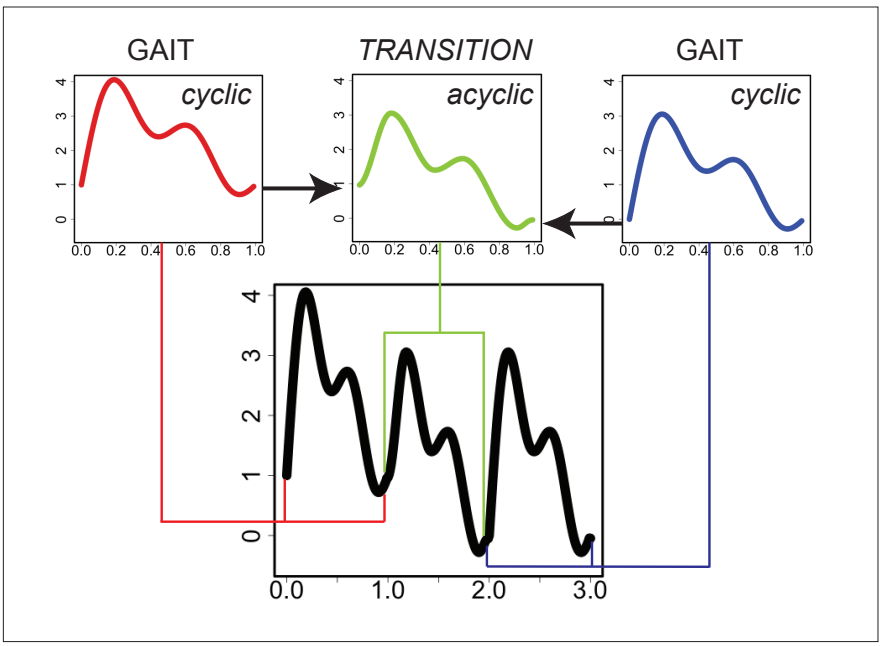


Figure 6.8: Acyclical piecewise representation connects neighbouring sinusoids with differing initial offsets.

The issue of acyclical data will be described further in Chapter 10 when the issue of gait transitions is discussed.

6.3 Chapter summary

In this chapter, the origin of motion data for use in the animation of animal models is explored.

In the first section, three separate potential sources of data are introduced namely photographic, motion capture and published. It is concluded that extracting data from photographs is time-consuming and inaccurate. Current methods for extracting motion data from video are immature and unreliable.

The use of video techniques is further explored in relation to motion capture. Although the animations that result from using motion capture data are often of very high quality, for the vast majority of animals, motion capture is prohibitively expensive or potentially impossible.

The final data source explored is the data published in biological journals, which is freely available and easily extracted. It is concluded that although this data is not directly usable for animation purposes due to its numerous varying sources and potential incompleteness, the data can be used as the basis for motion data generation and optimisation.

In the second part of this chapter, several representations for this data are presented. The use of motion data in a raw numerical form is discussed in advance of giving details of two alternative representations specifically targeted at motion data optimisation.

The first representation presented utilises a summation of sinusoids to represent the cyclical motion of a bone during a gait cycle. The second piecewise representation involves a segment by segment description of a curve, which is well suited to representing acyclical data.

Both of these representations will be discussed again in Part III when a series of experiments are presented which use the models described in Chapter 5 and the data described in this chapter to automatically produce animal animations.

Before these automatic methods are presented however, in the final chapter of this part, we present manual attempts at motion data generation for both kinematic and physics-based animations.

Chapter 7

Manual motion data optimisation

In this chapter, two manual motion generation systems are presented; one for kinematic animation and another for physics-based animation. In advance of the evolutionary computation-based automatic motion generation approaches that are presented in Part III, in this chapter, the precursors to those systems are discussed.

The manual motion generation systems described in the following sections not only allow for the development of basic motion data, but they ultimately provide the motivation to develop automated techniques. The physics-based system includes a combination of visualisations and motion editing tools, providing all that is required to create motion for the physics-based horse model described in Chapter 5. The kinematic animation system in contrast can be viewed as a simple tool to aid in the understanding of the relationship between a single joint's flexion and an entire limb's motion.

7.1 Kinematic CMA

The Curve Modifier Application (CMA) is a visual, interactive program that enables a user to see the kinematic motion produced in a limb from user supplied data.

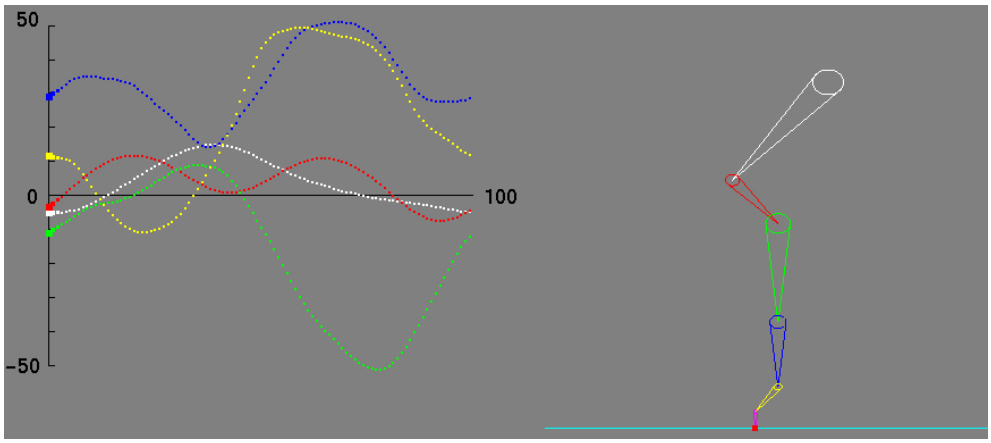


Figure 7.1: Curve Modifier Application (CMA): screenshot shows the bone motion curve visualisation (left) and limb animation (right).

A screenshot of the CMA is shown in Figure 7.1. The application displays the output animation for a single limb alongside a visualisation of the bone rotation curves.

Each curve has a colour that relates to the colour of its corresponding bone. As the system steps through the gait cycle motion data, the limb is animated and an indicator travels along the length of each curve, indicating the current stage of the gait cycle.

The user selects a single limb to be animated, which is constructed from the quadruped file format described in Appendix Section B.1.2.

Motion data files in the summation of sinusoids representation can be input to the application. The corresponding motion curves and animation will be displayed for the selected limb. To alter the motion, the

input data file must be manually modified and the changes can be viewed in the application.

A more interactive motion adjustment method is possible if the input data is in the piecewise representation. When data in this form is input, it can be modified in real-time as the animation is running.

7.1.1 Interaction

When using a piecewise representation, the user can select a curve to be adjusted and then modify it as the animation is running.

To adjust the data, a particular feature of the piecewise description is first selected. The value and position of that feature can then be dynamically changed. This process is illustrated in Figure 7.2.

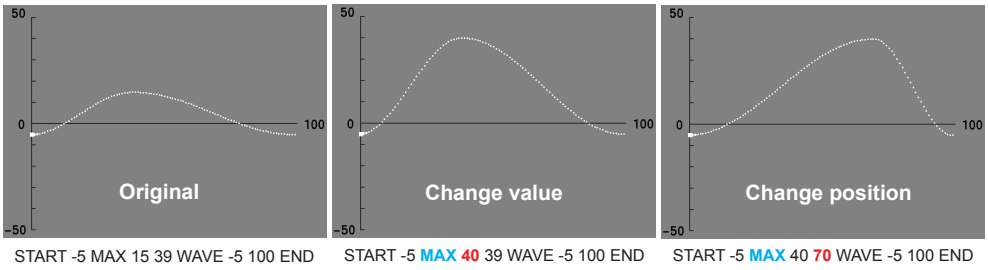


Figure 7.2: The leftmost image shows a single motion curve and its piecewise description. The other images show a value change (centre) and position change (right) for a selected feature.

In the example shown in Figure 7.2, only a single motion curve is displayed and the piecewise description of the original curve is shown underneath the leftmost image. Using specific keystrokes, the user selects a feature to change. In this case the first and only MAX feature is selected and its value is changed.

The effect of this value change can be seen immediately in both the motion curve (centre image) and the animation (not pictured). The user then makes another modification to the position of that same feature. This has the effect of elongating the curve segment that slopes up to this MAX point, as can be seen in the image on the right.

This dynamic modification of the motion data is relatively simple when using a piecewise representation. A similar approach with the sinusoidal representation would be less predictable for the user, as there would potentially be a larger number of parameters and it would be very difficult to intuitively modify a single section of the motion curve.

Regardless of representation however, manual motion adjustment can be a complex problem and the CMA provides visualisations to aid the process.

7.1.2 Visualisations

The main visualisations provided by the CMA are the motion curves and the limb animation, which are shown in tandem.

Although it is difficult for a user to focus on both visualisations at the same time, the animations repeat indefinitely and the relationship between the motion curve and the model limb's motion can become apparent within a few gait cycles.

In Figure 7.3, the relationship between a motion curve and a bone's rotation is illustrated. The set of images show the state of both the curve and the limb at six successive points in a gait cycle.

Prior to the commencement of the animation, the motion data corresponding to each bone in the selected limb is converted to a sequence of

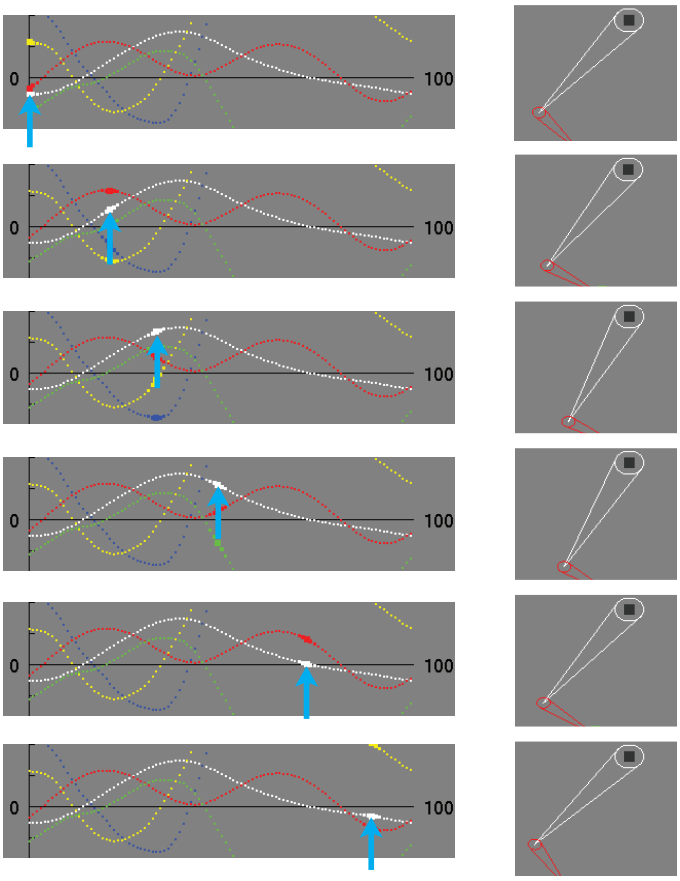


Figure 7.3: The relationship between a motion curve (white) and a bone's rotation is displayed. The blue arrows indicate the current point in the gait cycle.

100 discrete values. Each value represents the bone's angle at a point in the gait cycle. When the animation is started, the system steps through these values and the bone is rotated accordingly. The motion of the scapula of a horse's forelimb is shown in Figure 7.3.

The other visualisation provided by the CMA is the hoof motion bounding box. It can be difficult for a human to visually comprehend the region in space that a hoof moves through while it is moving quickly. To address this issue, the hoof leaves a fading trail that clearly shows the path of its motion. This trail can be seen in Figure 7.4.

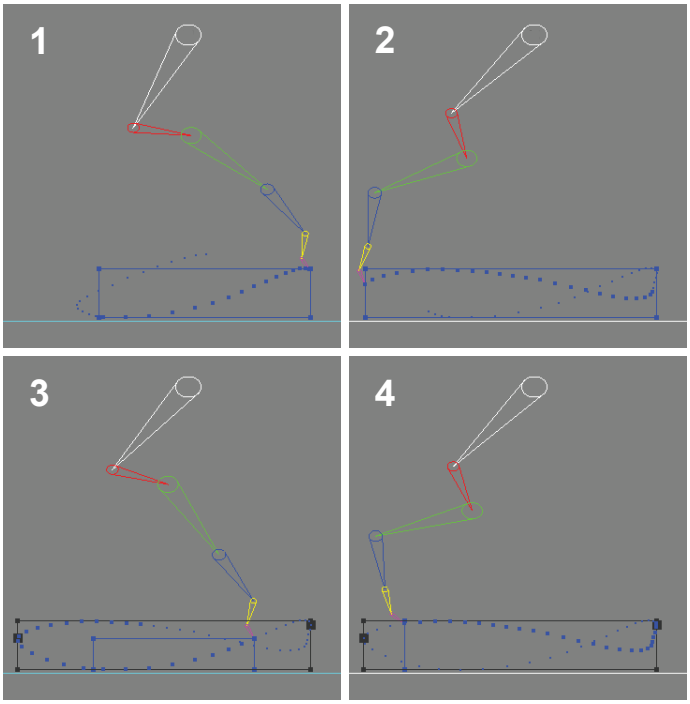


Figure 7.4: The sequence of images shows the CMA's dynamic bounding box as it is established at the first gait cycle. A fading trail following the hoof's motion can also be seen.

Also shown in Figure 7.4 is a bounding box. During animation, the limb itself does not translate horizontally or vertically. This allows the motion of the hoof to be bounded.

In image **1** and **2**, the limb is moving for the first gait cycle. As the hoof moves for the first time, a bounding box is expanded to the limits of the hoof's motion area. As the animation moves to the next cycle, the blue bounding box is coloured black and remains static, as defined by the last cycle. For this current gait cycle, a new blue bounding box is expanded, as shown in image **3** and **4**.

This visualisation is of use when modifying motion data dynamically as the change that a particular adjustment makes to the hoof's movement can be instantly compared to the previous cycle.

7.1.3 Applications

The CMA is a convenient way to visualise motion data and see the effects of certain modifications in real-time.

For producing a kinematic animation, the CMA can quickly enable an animator to produce recognisable animal motions. The manual generation of motion using this system is based on trial and error however, and the results, while aesthetically acceptable, are not necessary realistic.

In addition to the features described in this section, the CMA has been extended for use with an automatic kinematic motion gait generation system, which will be discussed in Chapter 10.

For the manual production of realistic animal motions, the data modification options offered by the CMA are limited. In the following section, a more fully featured motion data development application for physics-based animation is described.

7.2 Physics-based MDDE

In this section, the Motion Data Development Environment (MDDE) is presented. Designed primarily for physics-based animal animations, a user can edit motion data and visually evaluate the resultant animation in real-time. When creating physics-based animal animations, using motion data measured from real-life animals in motion can increase the realism of resultant animations.

As discussed in Chapter 6, this motion data may not have been measured from an animal with proportions matching the computer constructed model. In addition, the data may be noisy as a consequence

of the original measuring process or the extraction-conversion operation. The motion data must therefore be adjusted for use with a particular model and the MDDE enables a user to do this in an intuitive manner.

By displaying the data and animation in tandem, the relationship between the motion data and the resulting motion is emphasised. A range of visualisations also helps a user to understand the effect that a particular adjustment of the data has on the model's motion.

MDDE component overview

The MDDE has three main elements; the animation display, the data visualisations and the interface, as shown in Figure 7.5.

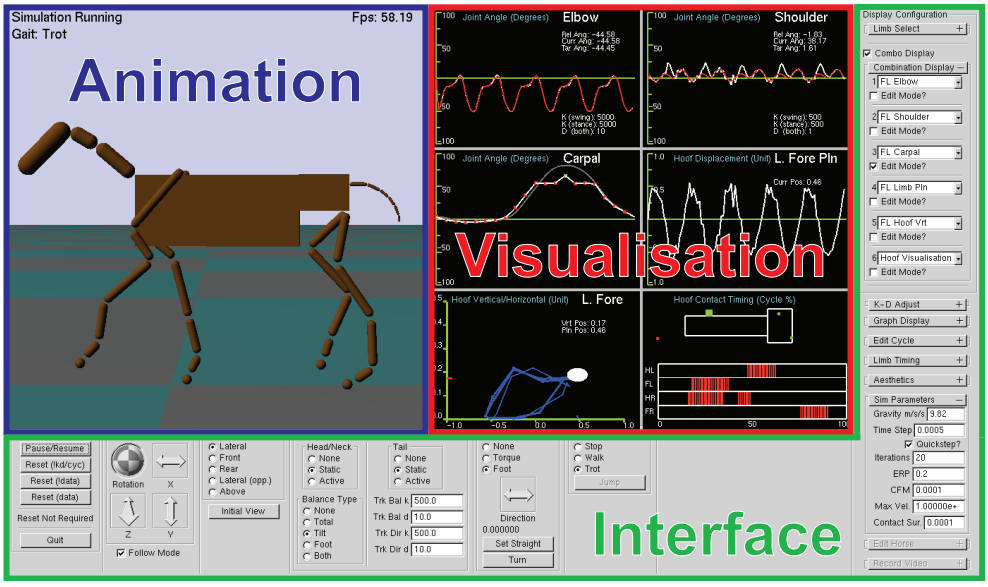


Figure 7.5: The Motion Data Development Environment (MDDE) with the animation, visualisation and interface components clearly marked. *Video 7.1 Motion Data Development Environment*

The animation displayed in the MDDE is created with the physics-based animal animation technique described in Sections 2.3 and 5.3, using OpenGL (Section 5.2.1) and ODE (Section 2.3.1).

The visualisations are also created using OpenGL and are discussed in Section 7.2.3. The application's interface is constructed using the OpenGL User Interface Library (GLUI) which is easy to use, OS independent and provides all of the MDDE's controls [162].

7.2.1 Interface

In this section, aspects of the MDDE's interface are described.

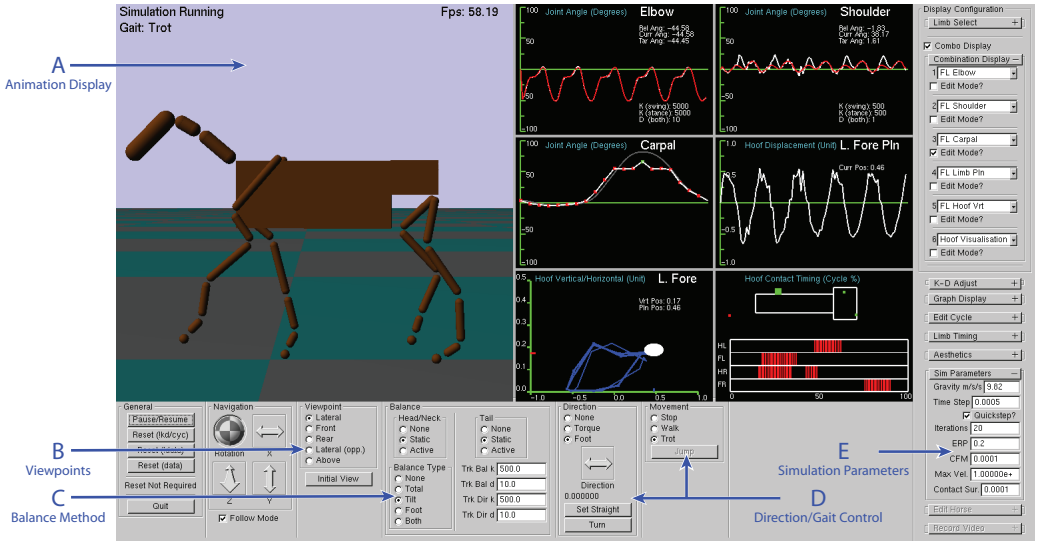


Figure 7.6: Annotated screenshot of the Motion Data Development Environment (MDDE).

The interface gives the user control over the viewpoint, the model's movement and simulation parameters. A screenshot of the MDDE is shown in Figure 7.6 and the labelled features are described below.

A The animation display shows the model's movement in real-time.

B There are several dynamically selectable predefined viewpoints.

C Model features such as balance technique can be switched in real-time.

D The gait and direction of the model can be controlled.

E ODE simulation parameters can be adjusted through the interface.

Other aspects of the animation can also be modified in real-time through controls on the interface panels.

The rotation sphere and arrow buttons on the interface's navigation panel provide full 3D navigation of the scene. As a user examines the model in motion, the input spring-damper coefficients can be modified for each joint through the interface, as will be discussed in Section 7.2.4.

In terms of motion, the gait pattern of an animal can be selected from those specified in an input file. The interface also allows a user to manually adjust the phase differences between limbs. In this manner, the effect of novel gait patterns can be explored.

The most important feature of the interface however, is its facility for letting a user configure the motion curve display and editing windows, which are described in the following section.

7.2.2 Animation and motion data editing

As described in Chapter 5, the physics-based horse model is constructed from input skeletal data using the ODE primitives, and displayed using OpenGL.

A list of the input data files required by the MDDE is provided in Appendix Section B.3.1. The animation process begins when a gait pattern is selected via the interface and the user initiates the simulation. When the simulation commences, the input motion data pertaining to the selected gait is loaded.

This motion data is supplied in the discrete value representation discussed in Section 6.2. As each bone's motion data only consists of the 20 data points sampled from the published joint-angle plots, a cubic B-spline interpolation technique is employed.

In mathematics a spline is a function that is defined by polynomials in piecewise manner. A basis-spline or B-spline is a generalisation of a Bézier curve (a type of parametric curve) [161]. The interpolation of the data points is shown in Figure 7.7.

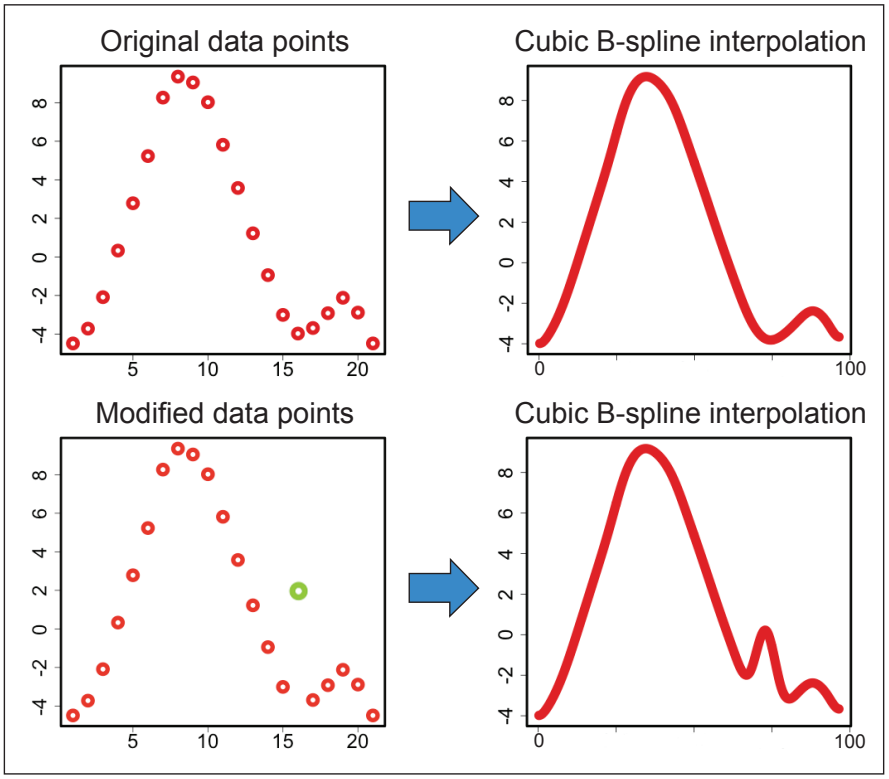


Figure 7.7: The images on the left show a plot of some data points. The images on the right show a cubic B-spline interpolation of those points. The effect of editing a data point (larger, green point) can be seen in the resultant interpolation.

The images on the left of Figure 7.7 show the data points of an input motion file. In the lower left image, a point has been selected and modi-

fied (the larger, green point). The effect on the interpolation can be seen in the bottom-right plot.

By interpolating between the data points, the data for a gait cycle is represented by 100 values instead of 20. This allows for a smoother animation as each of these data points represent a bone's rotation at a particular point in a gait cycle.

Using a cubic B-spline interpolation ensures that even after a significant modification of the data points, a smooth discretised motion curve is always produced. Once interpolated, this data can then be used by the motion controllers in the manner described in Section 5.3.2.

Motion editing

Once the simulation is running and the physics-based model's animation is displayed, the motion data can be viewed and edited in real-time.

In Figure 7.8, two visualisations of the motion data are presented. The visualisation shown at the top of the figure is a real-time updated plot of the motion curve for a selected joint, with other relevant data also displayed. This visualisation provides an abstract view of the attached bone's motion, complementing the animation display.

The MDDE's main purpose is to allow a user to modify the motion data and view the resultant animation in real-time. The motion data editing visualisation is shown in the bottom image of Figure 7.8.

Using the interface options, a user can select a motion curve to edit. As can be seen in Figure 7.8, the regularly spaced red dots represent the motion data points and the smooth grey curve is the most recent cubic B-spline interpolation of the data.

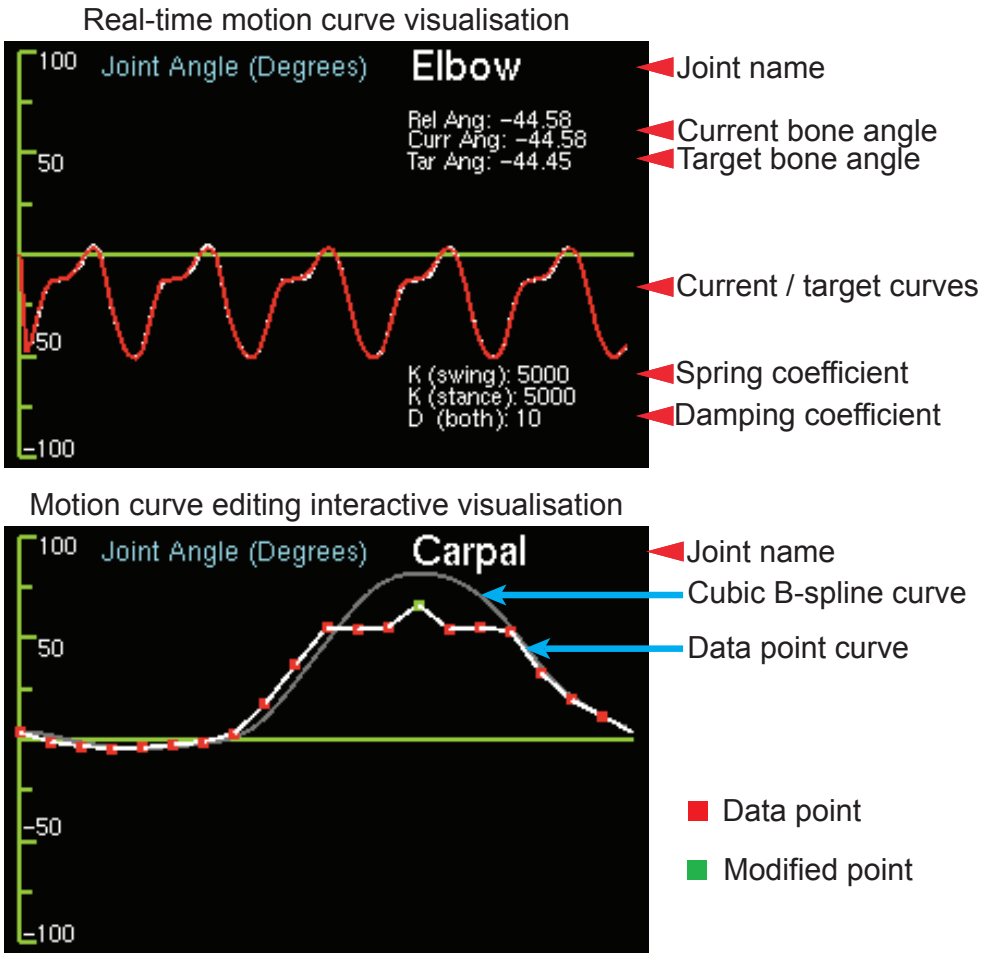


Figure 7.8: The top image shows a visualisation of a joint's motion curve. The bottom image shows the interactive motion curve editing window.

The user selects one of the red target motion data points to edit. The selected data point, which turns green, can then be dragged up or down to modify its value. When the data adjustment is complete, the data is re-interpolated and the adjustment's effect on the model's movement can be immediately observed and assessed.

Direction and balance

The systems for motion data generation presented in this thesis all pertain to locomotion of an animal in a straight line. The MDDE however, also bestows its quadruped model with basic direction control. A user can request that the model turn left or right by a specified angle. In response, the model moves an appropriate limb, during its ground contact phase of the gait cycle, away from the midline of the body while maintaining contact with the ground.

The model rotates in response to the turning force which has propagated through the limb from this pushing hoof. The abducted limb is then moved back towards the midline during the transfer phase of its gait cycle. This method of direction control is reasonably effective and mimics the natural turning technique of a real horse [81].

The model can also use a combination of balancing techniques including foot placement and minor correcting forces. Balance is always an issue for physics-based models, as alluded to in Section 2.4.2, but one that is not addressed in this thesis. Generated motion data is expected to have some inherent stability, however, during sustained locomotion the model will fall without a balancing system.

A very basic foot placement-based balancing technique is implemented and selectable in the MDDE. Whilst effective considering its simplicity, it is in no way a substitute for a robust approach to balance. As such, the physics-based horse models presented in this thesis are balanced through the application of minor corrective forces throughout the gait cycle.

7.2.3 Visualisation and verification

Visualisations of motion data can allow a user to examine a model's movement in great detail.

While a visual analysis of an animal animation can give an indication as to whether the model is moving in a realistic manner, it is also important to closely examine individual aspects of the motion. In the MDDE, this is made possible through the provided motion data visualisations.

The most obvious visualisation is the animation of the horse itself. Its stylised appearance clearly shows individual bone movement alongside the other motion data visualisations.

A user can configure the visualisation window to display a combination of six different visualisation types namely: motion curve, motion curve edit mode, limb reach, hoof lift and hoof timing/contact. A combination of these visualisations can be seen in the screenshot shown in Figure 7.6.

When a user edits the shape of a motion data curve as discussed in the previous section, features of motion such as stride length and duty factor will be affected. As the sequence of rotations in the articulated limb is changed, the motion path of the hoof will also change.

The visualisations shown in Figure 7.9 each relate to limb and hoof motion. The limb reach-visualisation plots the extent to which a selected limb reaches back and forth in real-time. This is useful for assessing how changes in the motion data affect the length of the model's step.

Another similar visualisation is also available that plots the lateral motion of a limb. This is useful when investigating directional and bal-

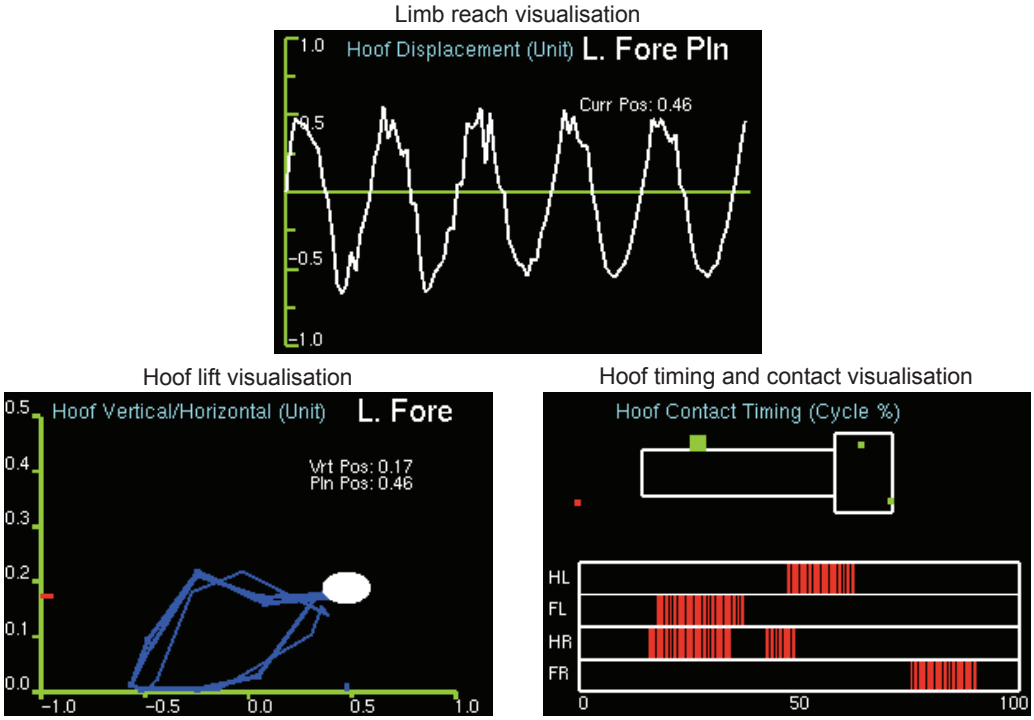


Figure 7.9: Three limb motion visualisations relating to limb and hoof motion.

ancing motion techniques. A third limb motion visualisation in the same vein, plots the vertical motion of the limb's end effector (hoof) and can be important for assessing a hoof's ground contact.

Hoof movement and contact

The quality of contact that the hoof makes with the ground is of high importance. From a visual inspection of an animation, it may appear that all hooves are making contact with the ground, pushing evenly as they are drawn back and then smoothly drawn forward for the next step.

Often while the general movement is correct, the hoof is in fact skipping along the ground. This skipping effect is wasteful of energy and introduces an unwanted bounciness in the model's gait. The dynamic

nature of the skipping also means that an individual hoof may exert forces that are disproportionate to the other hooves during a single cycle. This difference can introduce directional and balancing problems. The vertical and horizontal motion of a hoof can be monitored using the hoof-lift visualisation, shown in Figure 7.9, and adjustments to the motion data can be then made as necessary.

The position and timing of the hooves' ground contact also has a major affect on an animal's locomotion, direction and balance. The final visualisation shown in Figure 7.9 addresses the temporal aspect of the hooves' ground contact. Included in the hoof timing visualisation is an illustration that depicts the hooves' positions relative to the model's trunk and indicates when each hoof is in contact with the ground. A stick diagram displays ground contact information for the most recently completed cycle, highlighting exactly when each hoof made contact and the persistence of that contact.

In the following section, some successful applications of the MDDE are described, concluding this chapter.

7.2.4 Applications

The MDDE provides the tools that enable a user to manually assess and modify motion data.

Initial attempts at animating a physics-based horse model were difficult as data was unavailable. By creating a system in which even very raw data could be crafted into a sustainable gait, usable data was created through a process of continuous refinement. In this final part of the chapter, some of the successful applications of the MDDE are recounted.

One of the first challenges in manually producing motion data was the setting of the spring-damper coefficients, introduced in Section 5.3.3.

Spring-damper coefficients

Before the process of manually tuning motion data for the physics-based horse model could begin, the spring-damper (s-d) coefficients needed to be set for each joint in the model.

Unfortunately, without motion data it is difficult to test potential s-d values. Exploiting the MDDE's ability to restrict or isolate individual bones in each limb, a rough set of values were assembled which could rotate each bone and its connected bones through a simple sinusoidal rotation without causing instability in the simulation. The rough set of s-d values was then improved upon through use of the raw, untuned motion data.

The bone rotations measured from a real-life animal are more complicated than a simple sinusoidal rotation. The quick changes in direction of a curve caused the rough s-d values to be insufficient in some instances, and cause instabilities in others. The model could not move stably using the raw data so the s-d values were tuned for one limb at a time, while the rest of the limbs remained static.

A simplified view of the motion system has a set of springs pulling the bones of the model to target angles specified by the motion data. If a particular spring is not strong enough, i.e. its spring coefficient is set too low, the corresponding bone's motion curve visualisation will show a disparity between the bone's actual rotation curve and that of the target motion data.

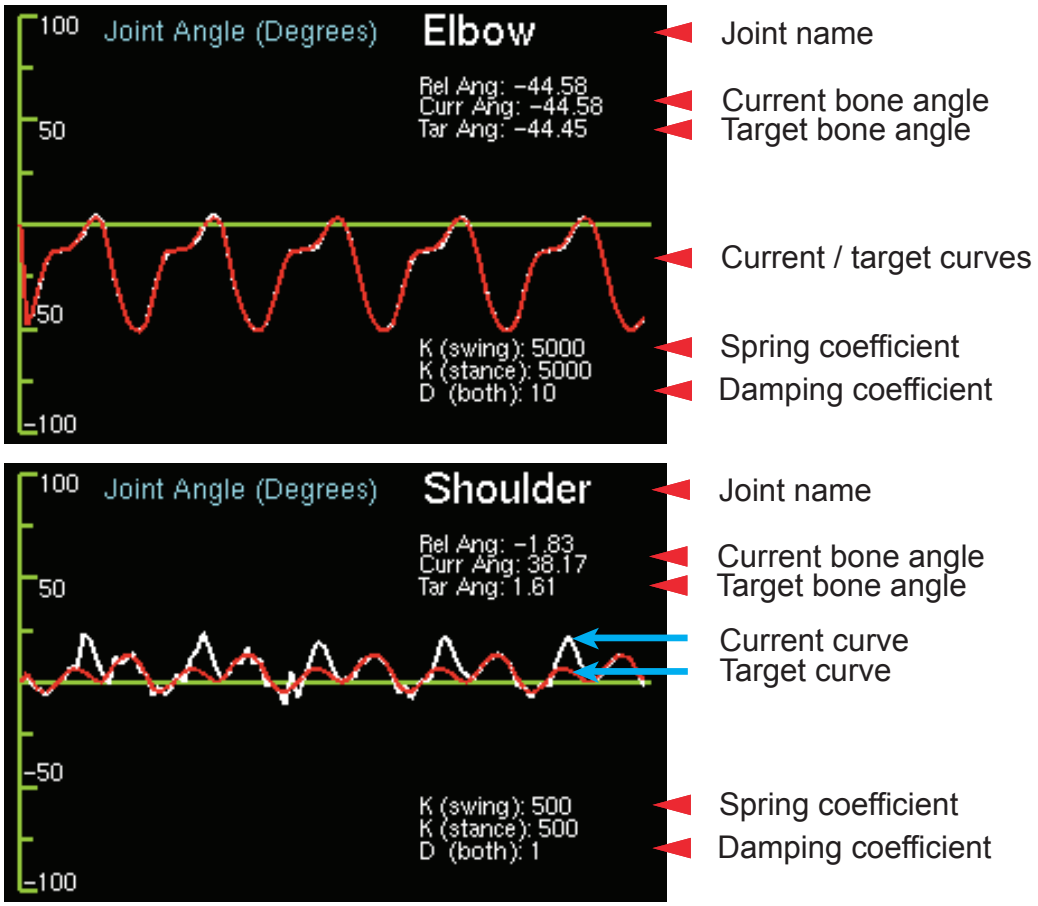


Figure 7.10: Two different examples of joint motion curve plots. The top visualisation shows that the target and actual angle of a bone are in synchrony. In constrast, the bottom visualisation shows disparity between the target and actual curves.

This problem is shown in Figure 7.10. The top "Elbow" visualisation shows the actual motion curve (white) and target curve (red) in synchrony indicating that the s-d values are set correctly for this joint.

In the bottom "Shoulder" visualisation however, the curves are out of sync indicating that the s-d values are incorrect. The erratic motion of the white curve indicates that the joint's spring forces are unable to control the shoulder's motion as it is subjected to ground reaction forces and the momentum and inertia of other bones.

When a spring coefficient is set too low, the result is often a lag between a bone's motion and the target motion. In some extreme cases, entire curves of the target data can be missed by the underpowered springs. One may be tempted to set the spring constants to a very high number to ensure this lag does not occur, however, this often results in the simulation becoming unstable.

Sometimes an over-tight spring attempts to shift a bone to its target position at such a rate that the damping factor of the equation cannot moderate the sudden increase in speed. A complicated combination of corrective spring and damping forces may be applied over the next few timesteps. These forces can accumulate and grow until the simulation explodes.

This was a constant issue with the manual s-d variable tuning process but using the motion curve visualisations it was possible to correctly tune each joint's s-d values for the horse model, one leg at a time. With this set of s-d values, the motion data could then be tuned. There were some additional adjustments made to the s-d values, as the model's locomotion process changed the dynamics of the system.

Ultimately a set of s-d coefficients were assembled that are used in the majority of the experiments presented in Part III. One exception to this will be seen in Chapter 9 where an automatic s-d coefficient generation system is presented.

Neck motion

The MDDE visualisations have also proved useful in detecting a correlation between the neck movement and skipping in the hind legs.

A static neck made the model front heavy. As a result the back hooves were skipping along the ground instead of thrusting the model forwards. In experiments where the neck was set to move in various ways, it was clear that the skipping was prevented if the neck reared backwards during the hind hooves' stance phase.

This neck movement shifts the centre of gravity of the model backwards and with more weight concentrated on the hindlimbs, the hooves made greater contact with the ground surface. This better contact increased the pace of the model, lessened the bounciness of the gait and improved direction control.

Ground interference

It was also observed that the model was not moving at the expected pace. Upon checking the hoof-lift visualisation, it became apparent that at the end of a limb's stance phase, as the hoof was being drawn up and forward for the next step, that hoof was impacting the ground surface.

This short contact with the ground and the associated friction was significantly reducing the model's momentum. The motion data was adjusted and the pace of the model increased to the expected level.

Basic motion data

Through use of the MDDE, motion data which describes a basic trot for the physics-based horse model was created. The process was tedious and time-consuming but the produced motion does resemble what is seen in nature and allows the horse model to move using only the forces generated by its hooves against the ground surface.

The ultimate result of the MDDE is a manually produced piece of motion data that, in this thesis, can be considered the benchmark to which the automatic approaches presented in Part III should be compared.

7.3 Chapter summary

In this chapter, the manual tuning of motion data is explored for both kinematic and physics-based horse models.

In the first section, a kinematic visual motion data editing system is presented. The Curve Modifier Application (CMA) allows a user to edit motion data whilst viewing animations of that data's motion curves and consequent limb movement.

The input motion data, in a piecewise representation, can be dynamically modified and the effect on hoof movement can be examined with the aid of a hoof motion bounding box visualisation.

In the second half of this chapter, the Motion Data Development Environment (MDDE) is presented. This system is primarily for use with a physics-based model, such as the horse model presented in Chapter 5.

The system offers sophisticated control over the physics-based animation and provides user-configurable visualisations, to assist in motion data modification.

The chapter concludes with a detailed description of how the spring-damper coefficients for the physics-based horse model are manually set. It is also noted that the MDDE was successfully used to manually tune motion data for the horse model.

7.4 Part II summary

The objective of Part II was to describe the construction and animation of the kinematic and physics-based horse models.

In the first chapter of this part, the construction processes were described. The origins of the data from which they are built was also discussed. OpenGL was introduced and the complexities of creating motion in a physics-based animal model using a spring-damper system was examined.

In the following chapter, the origin and representations of the motion data used to animate the horse models was discussed. The rationale behind the use of the two presented motion data representations will become more apparent in Part III.

The last chapter of this part presented the manual motion data modification systems. The kinematic system as presented is simplistic, however, it plays an integral part in an automatic gait and transition generation system which will be described in Chapter 10.

The motion development tool for physics-based animal animation is more richly featured. Using this animation system and its visualisations, the horse-model construction process was refined and a set of spring-damper coefficients was assembled.

The system was then used to develop motion data for a single gait, which allowed the horse model to move in a natural manner. The time and effort taken to manually produce this single piece of usable data however, motivates the need for an automated approach to motion generation.

In the next part of this thesis, the various experiments undertaken to automatically generate and optimise motion data using evolutionary computation techniques are presented.

Part III

Experiments with GE

Part III is the experimental section of this thesis. The presented experiments are principally concerned with automatically generating motion data for computer constructed horse models.

In each experiment, Grammatical Evolution (GE), a grammar-based evolutionary algorithm introduced in Chapter 3, is used to evolve motion data for a different animation application. Details of the GE grammar, fitness function and software system implementation are given in relation to each experiment.

In Chapter 8, a physics-based motion data optimisation system is presented. The use of different grammars for motion data optimisation and novel motion generation is described. Dynamic adjustments to the grammar and fitness function designed to speed up the evolutionary process are also presented.

Chapter 9 focuses on using the GE-based motion data optimisation system to animate quadrupeds of varying breed, age, conformation and even species using a single piece of source motion data. The first major experiment in this chapter describes an attempt to retarget motion data from one species to another using the GE optimisation approach and observations of natural evolution. Issues with this approach motivate a subsequent experiment, which describes how a model's spring-damper coefficients can be automatically generated as data is optimised for horses of different ages.

In the final chapter of Part III, a system is described in which realistic gaits and transitions for a kinematic horse model are generated. The generated motion data is used in an animation system, which gives a user full control over the model's motion in real-time.

Chapter 8

Automatic physics-based motion data optimisation

In this chapter, a GE-based system is presented that can be used to optimise motion data or generate novel motions for use with a physics-based quadruped model. The motion optimisation system and experiments presented in this chapter are published [134].

To restate the reason for developing such a system; physics-based animal animations require some measured animal movement data for realistic motion. This data is expensive to acquire through motion capture and inaccurate when extracted from images or estimated by an artist. Taking published motion data and manually tuning it to produce sustained motion in a physics-based animal model is laborious, as described in the previous chapter. Each of these factors motivates the need for an automatic approach to motion generation.

A grammar-based GP approach such as GE is employed to this motion data optimisation task as it allows the structure of the motion data

itself to be evolved, rather than simply optimising a set of parameters. Additionally, GE's separation of search and solution space allows complex problem domains such as animal motion, to be easily expressed and evolved through a grammar. For example, the summation of sinusoids data representation, introduced in Section 6.2.1, can be easily incorporated into a grammar, as will be seen in a later section.

The GE-based optimisation system and experiments presented in this chapter take motion data obtained from veterinary publications and optimise it for use with a physics-based horse model. Rather than simply performing a parameter optimisation on the data however, a Fourier analysis approach is used instead.

By representing the model's motion data as a summation of sinusoidal functions, the optimisation process operates on a more minimal set of parameters. Motion variation is provided through the concatenation of sinusoidal functions of differing amplitude and frequency. This representation is compact and mimics the sinusoidal nature of muscle movement. It may also give the evolutionary process more freedom to evolve than a parameter optimisation.

The experiments in this chapter are presented as follows. A variety of tests using different grammar types are explored in Section 8.2, including those that optimise motion data and those that freely generate motion. This is followed by an exploration of evolution speed-up strategies in Section 8.3 and a brief examination of an uneven terrain problem in Section 8.4.

Before this however, the GE system, grammar structure and horse simulation fitness function are discussed in the following section.

8.1 GE system overview

All of the experiments described in this chapter use a GE system for the evolutionary search and a physics-based horse model as the fitness function.

The specific GE system used is a Java-based implementation called GEVA [148]. GEVA supplies the characteristic GE genotype-phenotype mapper and also provides a GUI, through which evolutionary parameters can be controlled [146]. The EA search algorithm uses a variable-length integer encoding and further details can be found in the technical specification [147].

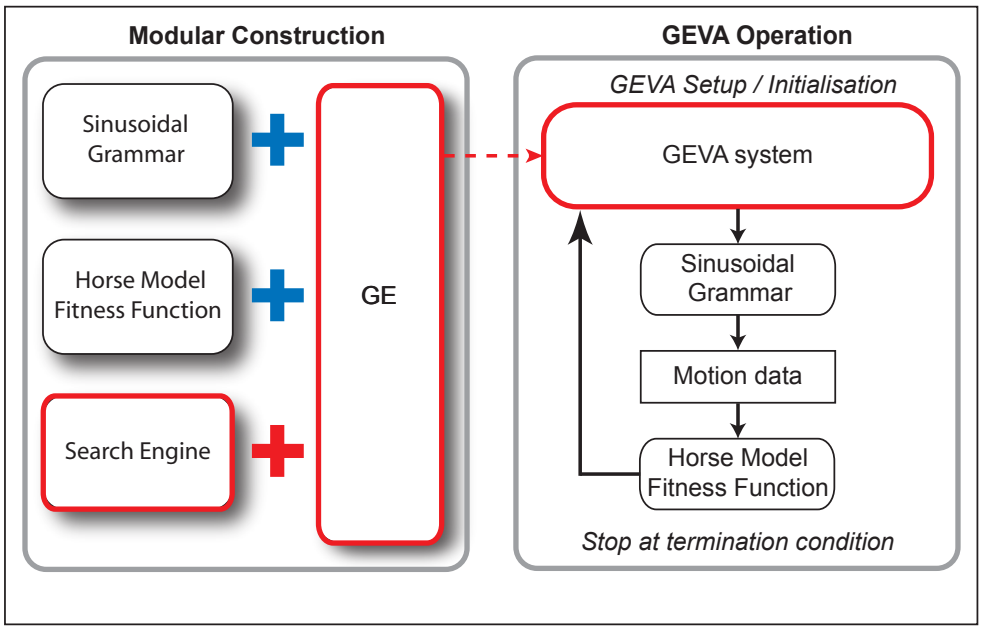


Figure 8.1: The modular construction of a GE system for motion data generation is shown in the image on the left. The implementation of such a system with GEVA is illustrated on the right.

A modular diagram of the GEVA-based motion data generation system is shown in Figure 8.1. The evolutionary search and GE genotype-

phenotype mapping process are both handled by GEVA. The fitness function however, is a separate physics-based horse model simulation application, to which the GEVA system passes potential solutions for assessment. The fitness function is discussed in detail in Section 8.1.2.

Each motion data optimisation process runs for a number of generations specified by the set of parameters passed to the GEVA system. The evolutionary search parameters used for each experiment in this thesis are presented in Table 8.1.

Table 8.1: GEVA parameters.

Parameter	**Value**
Generations	50
Population	75
Maximum wrapping	3
Replacement strategy	Generational
Elite size	7
Selection (size)	Tournament (3)
Initialisation	RampedFullGrow
Maximum depth	10
Grow probability	0.5
Crossover probability	0.9
Crossover point	Single point (randomly chosen)
Mutation probability	0.02

As previously mentioned, the motion data optimisation system presented in this chapter evolves data in the summation of sinusoids representation. The process by which the motion data and the general sinusoidal representation is incorporated into the grammar is discussed next.

8.1.1 Grammars

The grammars used in the GE mapping process must provide for the construction of syntactically correct motion data in the summation of sinusoids representation. The Backus Naur Form (BNF) notation and GE grammars in general were introduced in Section 3.5.2. In this section, the production of specific grammars for motion generation is described.

The aim of the experiments presented in this chapter is to automatically optimise a particular piece of motion data for use with a physics-based horse model. The motion data that is to be optimised, which will be often referred to as the seed data, is that which was extracted from the veterinary literature, as described in Section 6.1.3. This seed data must be incorporated into the grammar so all mapped solutions will essentially be modified versions of this original motion.

Once the seed data is incorporated into the grammar, the other biggest considerations in the grammar construction process is the manner in which the data is modified and the degree to which the mapped phenotypes are allowed to differ from this seed data. Depending on the grammar, phenotypes can be constrained to remain close to the seed data or allowed to deviate greatly.

The question of how constrained to make an optimisation can often only be answered through observation of several test optimisations. The manner in which the data is modified however, depends on the application and the representation.

The seed motion data is represented as a summation of sinusoids. The sinusoidal waveform is shown in Equation 8.1 as a function of time.

$$\langle amp \rangle * sin(\langle freq \rangle * 2 * PI * time + \langle phase \rangle) \tag{8.1}$$

The data could therefore be optimised by manipulating the variable values of its constituent sinusoidal functions. The values of the *amp* and *phase* parameters in each function could be optimised within some range of values defined in the grammar. As the seed data dictates the frequency parameters of these functions, the range of potential solutions is constrained.

This grammar-based parameter optimisation may therefore not allow for the generation of a full range of motions. To provide greater flexibility, the motion can be optimised by concatenating sinusoidal functions to the seed data from a fuller range of frequencies. Both this concatenating functions approach and the parameter "numerical" optimisation are compared experimentally in Section 8.2.1.

Grammar creation

A grammar's structure is determined by the problem domain, as the phenotypes created from the grammar are solutions to a specific problem. Creating a good grammar for a particular solution type is often nontrivial but is of critical importance.

In this section, the creation of a concatenating functions grammar is described. The grammar must be written in a way that produces phenotypes containing motion data for each bone in the model, in the summation of sinusoids representation.

The grammar needs to provide rules from which a complete set of summations of sinusoidal functions can be built, according to the GEVA determined derivation sequence. The exact form of the phenotype that should be produced is presented in Appendix Section C.1.1.

The phenotypic motion data file consists of a sequence of keywords followed by a single summation of sinusoids string for each movable bone in the model. The grammar must be written in a manner that will produce this complete file from every successful GE mapping operation.

An abbreviated example of an actual grammar is shown in Listing 8.1. Using the method described in Section 3.5.3, one can explore the possible derivation paths of this grammar.

The grammar presented in Listing 8.1 contains a set of function definitions, <function>, which have varying amplitude and frequency ranges. The values of these ranges are set based on the results of a Fourier analysis of the seed data described in Chapter 6. As a result of that Fourier analysis and subsequent data simplification, the seed data in a minimal summation of sinusoids representation is included in the grammar at the lines that define the <curve0> to <curve11> terminals.

It can be seen from the rest of the grammar that functions of varying frequency and amplitude can be added to or subtracted from the seed data expressions, producing the required phenotype.

Inclusion of the seed data, extracted from the published source described in Section 6.1.3, provides a template motion which the GE system attempts to optimise. It also reduces the search space associated with a multi-jointed animal model in comparison to a system that generates motion from nothing. The generation of gaits through grammars without

```
<prog> ::= <fcurve0> <newline> <fcurve1> <newline> ... <fcurve11>

<fcurve0> ::= <curve0> | <curve0> + <funcs>
...
<fcurve11> ::= <curve11> | <curve11> + <funcs>

<funcs> ::= <funcs> <op> <funcs>
          | <function>
          | <med_freq_amp_var>

<op> ::= + | -

<function> ::= <low_freq_amp_var> * sin( <low_freq_var> * 2 * PI * t )
             | <low_freq_amp_var> * cos( <low_freq_var> * 2 * PI * t )
             | <med_freq_amp_var> * sin( <med_freq_var> * 2 * PI * t )
             | <med_freq_amp_var> * cos( <med_freq_var> * 2 * PI * t )
             | <hi_freq_amp_var> * sin( <hi_freq_var> * 2 * PI * t )
             | <hi_freq_amp_var> * cos( <hi_freq_var> * 2 * PI * t )

<low_freq_var> ::= 1 | 2
<med_freq_var> ::= 3 | 4
<hi_freq_var> ::= 5 | 6 | 7 | 8

<low_freq_amp_var> ::= 0 | 0.25 | 0.5 | ... | 20
<med_freq_amp_var> ::= 0 | 0.1 | 0.2 | ... | 4
<hi_freq_amp_var> ::= 0 | 0.05 | 0.1 | ... | 1

<curve0> ::= 6.97+7.7*sin(1*2*PI*t+-1.07)+2.56*sin(2*2*PI*t+2.97) ...
...
<curve11> ::= ...
```

Listing 8.1: An example of a grammar based on the concatenation of sinusoidal functions to the seed data. Note the seed data at the bottom of the grammar. (Omitted terms are represented by '**...**').

seed data is explored later in this chapter. In the next section however, the physics-based horse model application is described in terms of its use as a fitness function.

8.1.2 Physics-based simulation fitness function

The fitness function in the GE motion optimising system uses a simulation of a physics-based quadruped model in motion. The Physics-based Quadruped Simulation (PQS) application takes an input motion data file, simulates a quadruped model's motion using that data for a few gait cycles, and returns a fitness score.

The horse model is constructed as described in Section 5.3, from the data presented in Appendix Section B.1.3. The manually set spring-damper coefficients (Section 7.2.4) are also provided. Other aspects of the simulation such as gait pattern, Froude number, number of cycles to run and fitness weights are set in advance of the optimisation process.

The PQS serves both as a fitness function and as an environment to animate animal models. During the optimisation process, the model's motion is not usually animated. Besides being unnecessary, the real-time nature of an animation and the resources involved slows down the evolutionary process by a huge factor.

When the simulation is running, the phase difference of the limbs is determined by the selected gait pattern and the stride frequency is based on the Froude number argument. The input phenotype only provides bone rotation data for each movable bone in the body, and this is what is assessed by the fitness function.

Fitness function overview

The aim of a fitness function is to provide a scoring system that can identify the phenotypes that may lead to an optimal solution. Equally important is the awarding of poorly performing phenotypes and false positives with a bad score to prevent them from reproducing.

The horse simulation's fitness scoring system is based on dynamic similarity predictions and energy efficiency. As discussed in Section 4.5, dynamic similarity theory states that a mammal travelling at a particular Froude number will share gait characteristics with other similar mammals travelling at the same Froude number.

A piece of motion data is optimised to move the model at a particular Froude number. From that Froude number predictions are made regarding gait pattern, stride frequency, stride length and duty factor (Table 4.2). The gait pattern determines the phase difference between the limbs and the stride frequency prediction determines the rate at which the limbs move during the fitness assessment.

The stride length and duty factor predictions are both used to score an input phenotype. A phenotype that exhibits stride length and duty factors that perfectly match the predicted values attain a score of zero, indicating optimality; the better the motion the lower the score. The score given to values that vary from the predictions is a function of the magnitude of the difference.

Energy efficiency of the motion data is also a factor in the fitness score. In nature, animal morphology and motion patterns have generally evolved to use minimum energy to travel a desired distance at a desired velocity [6]. The fitness function therefore rewards those phenotypes that use minimal energy. The energy calculation is described in a later subsection.

Each of the fitness components has an associated weight that can be adjusted by the user. Each component's contribution to the fitness score is a function of its weight and a measure of the error from its predicted value. Using this approach, a perfect score of 0 should never occur as the energy component is a weighted sum of the model's total energy use, averaged per cycle and adjusted to be in proportion with the other fitness components. As the model must expend energy to move, even the most optimal gaits will have a positive fitness value.

Dynamic similarity scores

Stride length is the distance between two successive placements of the same hoof during locomotion. A limb's duty factor is the percentage of a gait cycle in which a limb is in contact with the ground. Both of these gait characteristics are used to judge the optimality of motion data.

During a simulation, duty factor and stride length values are measured at every gait cycle, for each of the limbs. At the end of the run, average stride length and duty factor values are calculated for each limb and then separately for each pair of forelimbs and hindlimbs. The difference between these averaged pair values and the predicted values for that Froude number is then computed.

Each of the forelimb and hindlimb duty factor and stride length difference values are then normalised according to the predicted values in the range 0-10, as stated in Equation 8.2. Each fitness score is then calculated as shown in Equation 8.3.

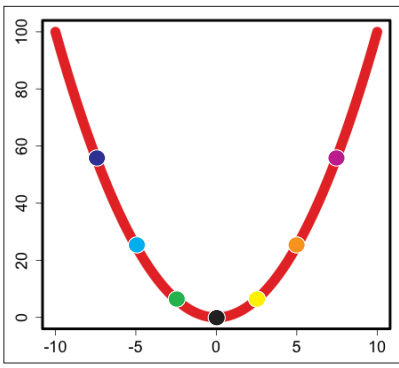
$$\text{normalised value} = (\text{difference value}/\text{predicted value}) \times 10 \tag{8.2}$$

$$\text{component fitness score} = (\text{norm fore}^2 + \text{norm hind}^2) \tag{8.3}$$

where *component fitness score* is the actual fitness value for the component and *norm fore* and *norm hind* are the normalised error values

calculated by Equation 8.2 for the forelimb and hindlimb respectively.

An example of the fitness calculation process is shown in Figure 8.2. To highlight the effect of Equation 8.3, several example fitness scores are plotted on the scoring curve (*left*). From this curve it can be seen that the farther a value is from its predicted value, the quadratically worse the score is. Any score that is at least double the predicted score is automatically given the worst possible score that the system allows.



Value	Predicted	Deviation	Norm.*10	Squared
0.25	1	0.75	7.5	56.25
0.5	1	0.5	5	25
0.75	1	0.25	2.5	6.25
1	1	0	0	0
1.25	1	-0.25	-2.5	6.25
1.5	1	-0.5	-5	25
1.75	1	-0.75	-7.5	56.25

Figure 8.2: The fitness curve used to score each fitness component is shown alongside an example of how fitness scores are calculated. Fitness weights are not included in this example.

The final step in computing the dynamic similarity portion of the final fitness score involves calculating a weighted value for each component. This weighting system allows a user to control the influence that each aspect of a model's movement has on the fitness score. The user can specify a different weight for each component of the fitness function and a component's *fitness score* is multiplied by this weight, as shown in Equations 8.4 and 8.5.

$$\text{stride score} = \text{stride length fitness score} \times \text{stride length weight} \quad (8.4)$$

$$\text{duty score} = \text{duty factor fitness score} \times \text{duty factor weight} \qquad (8.5)$$

In addition to these two dynamic similarity fitness components, the energy efficiency of the animal's motion is the final weighted component.

Energy efficiency score

In general, gait motion patterns evolve to allow an animal to move at a certain speed, using a minimal amount of energy [183].

The horse model fitness function exploits this observation, and for each gait cycle in a run, the energy used is calculated. The calculation is partly inspired by a GA approach to robot gait generation [100]. The presented fitness function awards high scores to the most energy efficient gait motions generated. This scoring scheme works with a high degree of success for a robot and the authors also observed emergent behaviour similar to that of real-life quadruped mammals.

To calculate the energy efficiency of a model, certain items of data must be tracked. The dynamic torques required to move each limb must be calculated at every timestep, along with the static torques required to support the body and prevent collapse. These values can be obtained by calculating the kinetic and potential energy of every bone in each leg.

In the horse model fitness function, this process is simplified. To calculate the energy expenditure of a single bone at each timestep, the

magnitude of the torque applied to that bone is multiplied by the angular distance that the bone moves over that timestep. Each limb's energy expenditure is calculated as the sum of the energy used to rotate each of its component bones. The energy expended by the entire model is the sum of the energy use of each of its limbs.

As the torque values used in this method's calculations do not relate to real-life values, it is impossible to compare them to a realistic cost of transport value. It is also nontrivial to calculate what the minimal amount of energy required to move the physics-based model might be.

The implemented approach calculates the "minimal" energy required to move the model as a function of the model's mass and a numerical constant. This constant is chosen to yield an energy value that is slightly less than the energy use value produced by the model, when moving with the most optimal motion data available.

This energy component of the fitness function is calculated using Equation 8.6.

$$\text{energy score} = \text{energy used} \times \text{energy weight} \tag{8.6}$$

Once all three components of the fitness function are calculated, a phenotype's actual fitness score is the sum total of each of the weighted fitness components as shown in Equation 8.7.

$$\text{fitness score} = \text{stride score} + \text{duty score} + \text{energy score} \tag{8.7}$$

8.2 Motion data optimisation experiments

In this section, several different motion data grammars are compared.

Grammars that utilise seed data are explored as well as free-style grammars that do not. In the case of these unconstrained, free-style grammars, the goal is not to produce aesthetically realistic gaits, but to test the capabilities of the multivariable fitness function, create novel movement and investigate how differing levels of domain knowledge affect gait generation.

The experiments presented in this section are divided into two categories. In Section 8.2.2 and 8.2.3, experiments involving the free-style grammars are presented. Before this however, those experiments which do use seed data are discussed.

8.2.1 Seed data grammar

In Section 8.1.1, grammars that contain seed data in the summation of sinusoids representation were introduced.

The question of whether a simple parameter optimisation of the seed data would allow for the generation of a rich variety of motions was raised. In response to this, an optimisation approach based on the concatenation of sinusoidal functions to the seed data was introduced.

In this section, these two approaches are experimentally compared. Each approach is described separately, starting with the parameter "numerical" optimisation approach.

Numerical approach

As parameter optimisation approaches are known to perform well for motion optimisation problems, a basic optimisation of the seed data's parameters is performed.

The first step in the optimisation process is to create an appropriate grammar into which the seed data is incorporated. Each term in the compact summation of sinusoidal functions is represented as a triple <amplitude, frequency, phase>. For each term of a specific frequency, both the amplitude and phase values may be optimised within a range of 25% of itself.

In certain cases the cyclical nature of the motion may be compromised by alteration of the amplitude and phase values, however, this usually results in a poor fitness score and on rare occasions, simulation instability. Any phenotype whose motion causes the simulation to become unstable or causes the model to flip, fall or move vertically over some threshold value, is immediately awarded the worst possible score.

The performance of the numerical approach will be discussed in Section 8.2.1. Prior to this, the concatenating functions approach is described.

Concatenating functions approach

To provide greater flexibility in terms of the range of motion that can be generated from the seed data, a concatenating functions approach is examined.

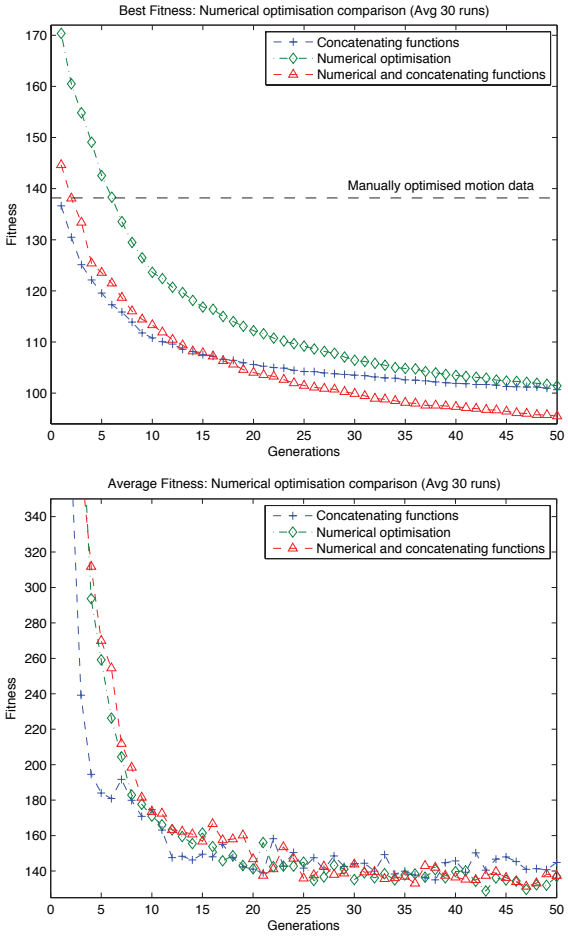


Figure 8.3: A numerical optimisation approach is contrasted with the concatenating functions grammar and a hybrid of the two approaches. Best and average fitness (averaged, 30 runs) are presented on the top and bottom respectively. *(Note difference in scale.)*

The grammar presented in Listing 8.1 optimises seed data by adding (or subtracting) sine and cosine functions of differing amplitude and frequency to the seed data. The frequencies of the appendable sinusoidal functions and the seed data summations have a range of 1 to 8Hz. Fourier analyis of the seed data shows that higher frequency functions have amplitudes less than the chosen threshold value of 1. These functions are considered less influential to the overall motion and are discarded for

compactness. The grammar also constrains appended functions of a particular frequency to have an amplitude of a specific range. This range is also based on observations from the Fourier analysis.

A combination of the numerical and concatenating functions approach is also experimented with. The parameters of the motion data are optimised as discussed in the previous subsection, whilst sinusoidal functions are concatenated to it. The results of all three of these approaches are presented next.

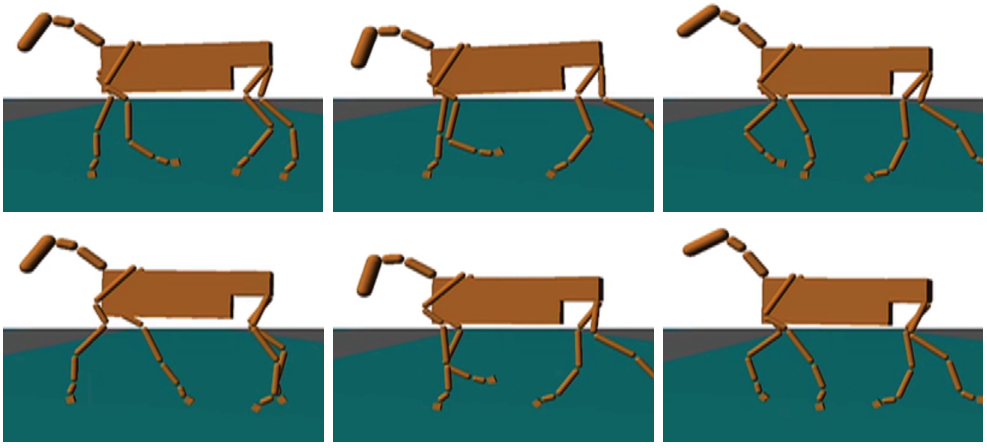


Figure 8.4: A sequence of screenshots showing the horse model moving with a trot. This motion was evolved using the concatenating functions grammar. (Motion sequence progresses from top-row, left to right and then bottom-row, left to right.)
Video 8.1 Published motion data (optimised)

Results

The best fitness plot in Figure 8.3, shows the numerical optimisation starting off worst. Gradually it improves and achieves a similar optimal solution score to the concatenating functions grammar. The overall winner in terms of best fitness is a grammar that uses a combination

of parameter optimisation and concatenating functions. The sinusoidal functions are added to the seed data, whose parameters are optimised in parallel.

As expected, the numerical approach performs well, as the seed data has been measured from an animal with very similar morphology to the model. In this case, the constrained nature of the numerical optimisation is perfectly acceptable. In this thesis however, one of the goals is to take a single piece of motion data and use it to generate multiple motions for differing models. For this scenario, the scope for deviation provided by the concatenating functions approach appears to be more appropriate.

In general, all three approaches excelled in this experiment. Although the motion of the majority of the evolved solutions remained close to the seed data, in all cases the motion data is subtly modified to provide a smoother motion that adheres to the dynamic similarity predictions.

A sequence of screenshots showing motion optimised using the concatenating functions grammar is shown in Figure 8.4. The motion data produced by each of the approaches described in this section appear visually realistic as they are based upon data measured from a real-life animal. In contrast to this approach, the free-style grammar experiments that do not use seed data are presented next.

8.2.2 Free grammar

The free grammar presented in Listing 8.2 contains neither seed data or sinusoidal function templates. This implies that the grammar includes no information on animal motion and does not suggest any constraints based on the animal's joint limits and muscle distribution.

```
<prog> ::= <fcurve> <newline> ... <fcurve>

<fcurve> ::= <expr>

<expr> ::= <expr> <op> <expr>
      | (<expr> <op> <expr>)
      | <pre-op> (<expr> * t)
      | <var>

<op> ::= + | - | / | *
<pre-op> ::= sin | cos

<var> ::= -20 | -19.75 | -19.5 | ... | 19.5 | 19.75 | 20
```

Listing 8.2: An illustrative example of a free-style grammar. The t variable is required by the simulation application so that generated motion data may be a function of time. The *fcurve* terms are omitted and represented by '**...**' for each movable bone in the model. Similarly the full range of *var* values is not shown.

As such, the motions resulting from this free grammar vary greatly across the 30 runs completed. In some instances, the model moves utilising only its front or hind limbs. Other runs exhibit a sequence of sudden hops to move the model. On a few occasions, motion is produced by placing the limbs squarely under the animal's body and using a high frequency, small amplitude, back and forth motion to "vibrate" the model along the surface.

Examples of the hopping and shuffling motions are shown in Figure 8.5. The hopping motion involves the model thrusting itself forward from a slightly crouched position with its forelimbs. The hindlimbs are relatively uninvolved in the locomotion process. In nature, a horse's hindlimbs tend to provide the majority of the thrust during locomotion, accounting for the large muscles of the horse's hindquarters. As information regarding the animal's musculature is not even implicitly incorporated into the grammar, it is not surprising that the produced motion is not entirely natural.

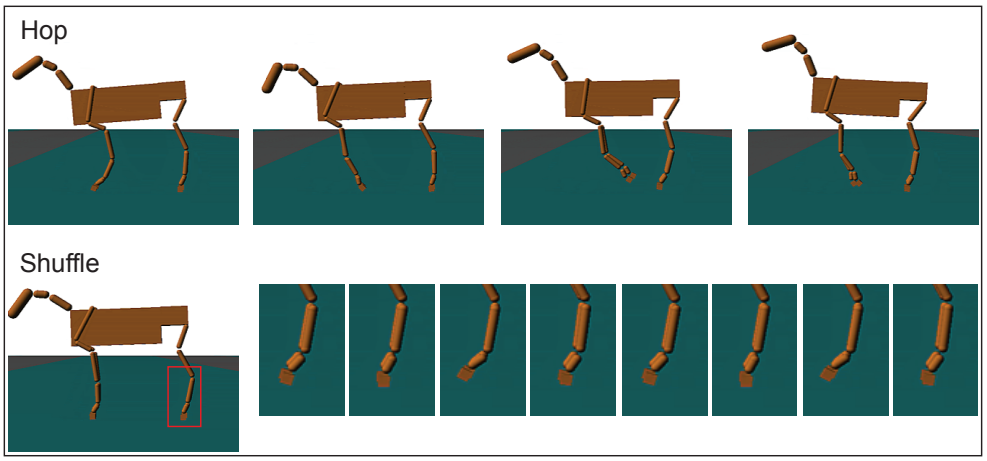


Figure 8.5: Two gait motions produced using the GE system and a free-style grammar. The top motion has the model moving using small hops. The lower images show a close-up of the model's hindlimbs moving with a shuffling motion.
Video 8.2 Free grammar hop
Video 8.3 Free grammar shuffle

The second distinct motion displayed in Figure 8.5 is the high frequency shuffling motion mentioned above. This motion translates the model quickly, however, it is physically implausible. The vast majority of the model's thrust is being produced by the flexion and extension of the forelimb and hindlimb fetlock joints.

As described in Section 4.3, the musculature of the horse's limbs is concentrated around the upper portion of the legs. The movement of lower joints such as the fetlock is controlled via very long tendons running down the back of the legs. It is unlikely that these tendons and the corresponding musculature could produce the forces necessary to move a horse in this manner.

The fact that these very different gait cycles achieve similar fitness scores demonstrates a flaw in the fitness function. In one scenario, disparate motions can score equally according to one or all the fitness func-

tion components. In another scenario, improvements in one aspect of the fitness score can overshadow other components, which suggests a more sophisticated fitness function is required.

The large variety of motions produced may imply that the ultimate shape of a solution using a free-style grammar may be randomly determined early in the evolutionary process. An interactive evolutionary computation technique could be employed in those early generations to guide the process towards a realistic motion.

In an attempt to avoid the more idiosyncratic motions produced by the free-style grammar, in the following section the exploration of grammars without seed data is continued, but with the inclusion of problem domain knowledge. Sinusoidal functions with frequency and amplitude values that match those observed through Fourier analysis of measured motion data are included in the grammar.

8.2.3 Sinusoidal grammar

The motion patterns produced using the free-style grammar are able to move the model at the correct velocity and in a seemingly efficient manner, however, this is without consideration for the physical ability and limitations of a real-life horse.

In an attempt to produce more realistic gaits without using seed data, the free-style grammar is modified to include sinusoidal functions. The grammar for this free-style summation of sinusoidal functions approach is similar to the concatenating functions grammar in Listing 8.1, except that the functions are not concatenated to any seed data.

Essentially, the knowledge that animal gait motion data can be decomposed into sinusoidal functions is incorporated into the grammar. In contrast to the free grammar in Listing 8.2, information about an animal's musculature is implicitly included through the specific frequency and amplitude ranges of the sinusoids.

Using this sinusoidal grammar, a total of 30 motion data generation runs were completed. While a small number of the generated gaits are visually unrealistic, the majority are comparable to the real-life motion of a horse. While not as realistic as those grammars which include seed data, there is potential for improvement given a more sophisticated fitness function and possibly increased population and generation values.

Notable results

Many of the motions generated closely resemble the motion of a real-life horse. There are two distinctive types of motion that are recurrent with this grammar, each of which is shown in Figure 8.6.

In **A** both the forelimbs and the hindimbs are moving together, as would be seen in an asymmetrical gait. What is interesting in this case is that the simulation is set to use a symmetrical trot limb-phase pattern. As limb pairs are using the same data, a movement such as this should be impossible.

It would appear that the motion data has evolved in such a manner as to overcome the imposed limb phase differences and move in an asymmetrical manner. Upon examination of the data, it appears that the evolution process exploited the fragility of the simulation's spring-damper coefficients. It does this by subdividing a single gait cycle into

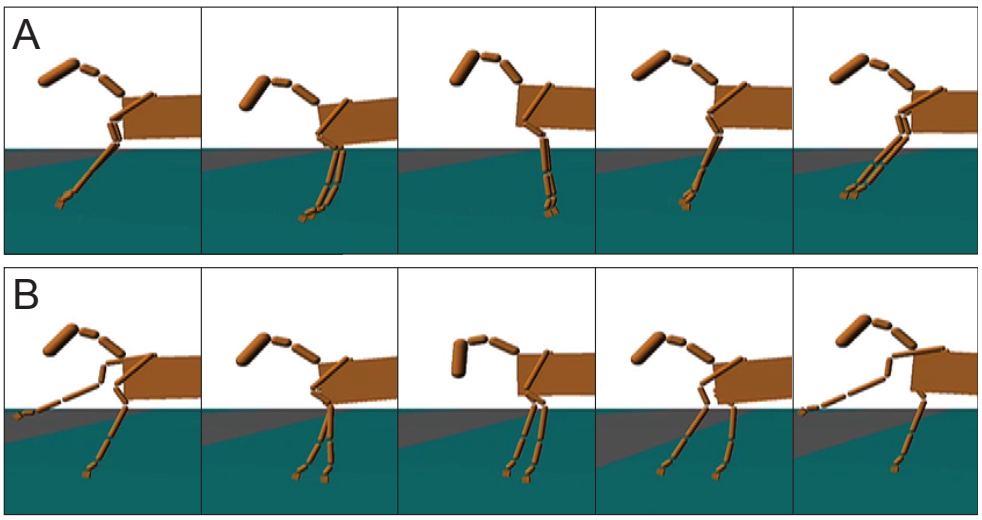


Figure 8.6: Two gait motions are shown created with the sinusoidal grammar (no seed data). In **A** the limbs are moving in synchrony whilst in **B** the limbs are moving alternately.
Video 8.4 Sinusoidal grammar gallop
Video 8.5 Sinusoidal grammar walk

two; rather then having a single transfer and stance phase per gait cycle, the evolved motion has two.

In a trot gait, the forelimbs (and hindlimbs) move at nearly a half a gait cycle apart. As the model begins to move from stance, the left forelimb is lifted from the ground and is moved forward in anticipation of its next grounding. The evolved motion data describes a second step in the same gait cycle, however, and as the right limb is about to move for the first time, the left forelimb begins to move for a second time.

The full weight of the front portion of the model's body is now on the right limb and the torque required to move the limb cannot be generated. As the left limb makes ground contact again, the motion controllers in the right limb attempt to catch up on their target motion data; they are lagging behind by a quarter of a gait cycle at this point. The right

limb is unable to catch up with its target data and instead settles into a pattern in which it moves in synchrony with the left forelimb.

The synchronous motion can be seen in Figure 8.6 **A**. This motion can be compared to another interesting motion generated through the sinusoidal grammar in **B**. In this case the model is moving in a symmetrical manner, however, the forelimbs move with a hugely exaggerated motion. Although this does not appear to be an efficient way to move, the large motions perhaps serve to balance the model somehow, which may result in velocity gains from the hindlimbs.

These observations are subjective and speculatory and a comprehensive analysis of this motion is nontrivial. The details of the evolutionary process are presented in Figure 8.7.

Sinusoidal and free grammar comparison

Figure 8.7 shows the free-style and sinusoidal grammar scoring comparably to the concatenating functions grammar in terms of fitness. This again illustrates the pitfalls of using a multivariable fitness function and few motion constraints. Out of the 30 free-style grammar runs, very different "optimal" solutions score similarly.

The free and sinusoidal grammars both produced a variety of interesting but very different and often unrealistic motions. It demonstrates that if realism is the goal, seed data, or constraints built into the grammar based on observations of animal motion, are required.

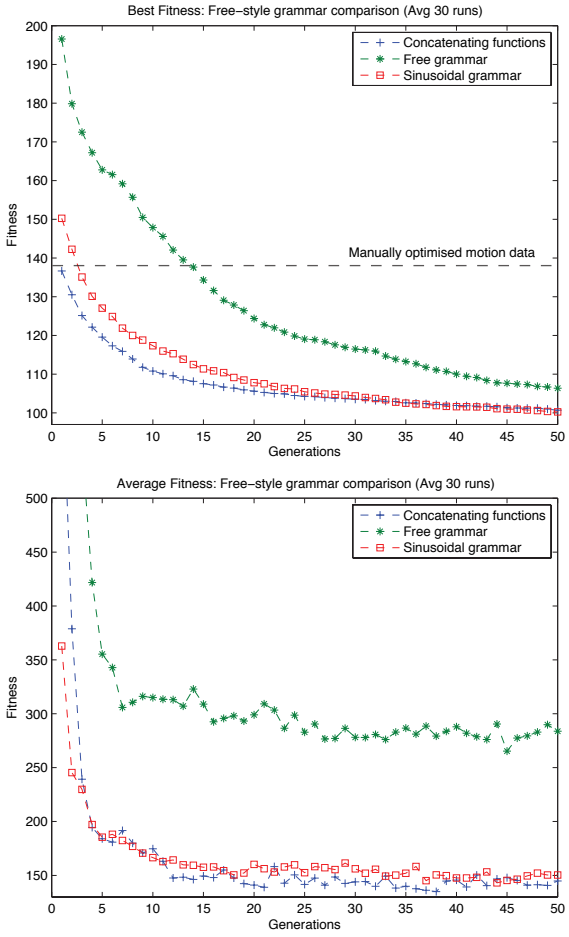


Figure 8.7: Free-style and sinusoidal grammars compared with the concatenating functions grammar. Best and average fitness (averaged, 30 runs) are presented on the top and bottom respectively. *(Note difference in scale.)*

8.2.4 Motion data optimisation conclusions

A comparison of the performance of each of the grammar types discussed in Section 8.2 is shown in Table 8.2. (Note that a lower fitness score indicates a better gait.)

The concatenating functions grammar improves upon the parameter approach in terms of fitness score, albeit by a small margin. The overall winner is a combination of the two. The concatenating sinusoidal

Table 8.2: Overall best and average fitness scores (averaged, 30 runs) achieved by each grammar alongside their respective standard deviations and standard errors. (Num. & Con. is the combination of Numerical and Concatenating functions.)

Grammar	**Avg. Best**	**Std. Dev.**	**Std. Err.**
Num. & Con.	95.3	5.2729	0.9627
Concatenating	100.333	8.5231	1.5561
Sinusoidal	100.4333	6.0039	1.0962
Numerical	101.3333	8.5715	1.5649
Free	106.3	19.05	3.4784

Grammar	**Avg. Avg.**	**Std. Dev.**	**Std. Err.**
Numerical	132.6506	37.6833	6.88
Num. & Con.	138.8273	57.2904	10.4597
Concatenating	140.9203	43.1542	7.8788
Sinusoidal	150.2595	44.2394	8.077
Free	279.1248	81.3519	14.8527

functions approach is found to be a compact method of representing and optimising motion data. Optimising seed data parameters whilst appending new sinusoidal functions provides the flexibility that may be required when retargeting motion data to other morphologies whilst maintaining realism as will be explored in the following chapter.

One significant observation from these experiments is that the multi-variable nature of the fitness function allows for significant motion variance between similarily scoring phenotypes. It may be benificial to use a multi-objective optimisation approach in future implementations, however, careful tuning of the fitness weights can still produce high quality results.

Every one of the grammars tested produces motion data that far outperforms the unoptimised seed data. A simulation of the model moving with the unoptimised data displays an extremely shaky motion. In some cases the model trips and bucks. After a few gait cycles, the balancing forces can no longer control the erratic motion and the model flips. In terms of fitness, the unoptimised seed data is awarded the worst possible score by the system as the motion is judged invalid.

Regardless of grammar, not a single motion produced using the described GE system causes unstable motion as that of the unoptimised data. While the motions produced by the free-style grammars may not be realistic, they at least allow the model to progress in a smooth manner. The motions produced by the grammars which do include seed data are judged in our opinion to be highly realistic and are a marked improvement on the manually tuned data described in Chapter 7.

In the following section, the effect of dynamically changing the fitness function and grammar during the evolution is investigated in terms of evolution rate.

8.3 Evolution rate experiments

The evolution of stable motion data takes many generations and the use of the physics-based simulation fitness function means that each fitness appraisal can take seconds to complete. Speeding up the evolutionary process is therefore of particular interest.

It is experimentally shown that modularly varying goals in an evolutionary system can greatly speed up the evolutionary process [98]. Ex-

ploiting this observation, two tests are presented in which the fitness function and grammar dynamically change in an attempt to improve the motion data evolution rate using the concatenating functions grammar.

In the first experiment, the weights of each component in the fitness function is changed as the evolution progresses.

8.3.1 Dynamic fitness function

For the varying fitness function test, the fitness weights are changed from generation to generation, as shown in Table 8.3. As can be seen from this table a fourth fitness component based on distance travelled is also included in these experiments.

Table 8.3: Fitness function weights during the fitness function variation run. The numerical value ranges in the top line of the table indicate the generations for which the listed weights apply.

	0 - 10	**11 - 20**	**21 - 30**	**31 - 40**	**41 - 50**
Distance	1	2	1	1	1
Duty factor	1	1	2	1	1
Stride length	1	1	1	2	1
Energy	1	1	1	1	1

The goal of the dynamically changing fitness function is to speed up the evolutionary process, prevent the process from becoming stuck at local minima and produce a more "well rounded" solution, i.e. one which optimises aspects of velocity, duty factor and stride length equally.

The results presented in Figure 8.8 do not show any speed-up. The change in fitness function does seem to drive the evolution forward in some situations. In this example, the change in fitness function causes a

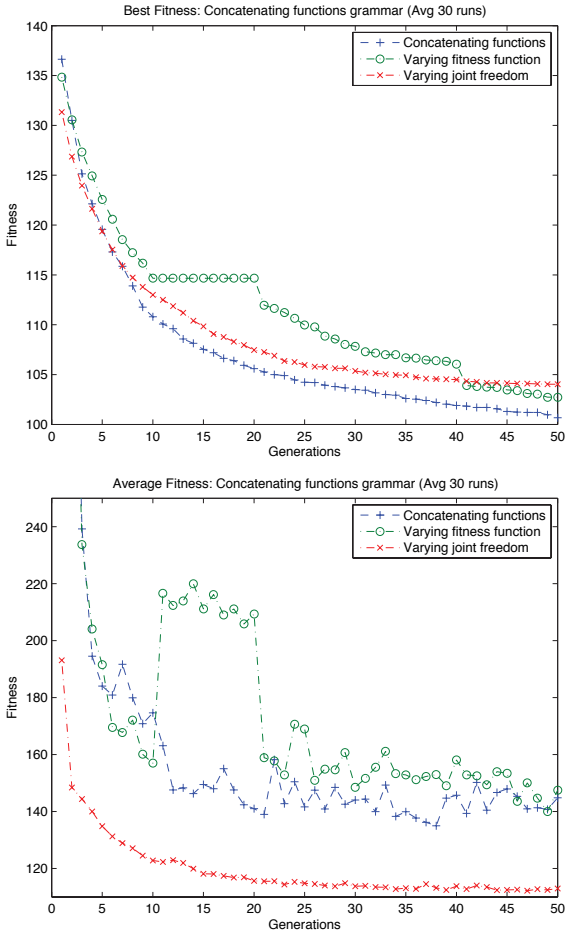


Figure 8.8: The concatenating functions grammar with alternating fitness and grammar strategies. Best and average fitness (averaged, 30 runs) are presented on the top and bottom respectively. *(Note difference in scale.)*

plateau in the best fitness score from generation 10-20. The large spike in the corresponding average fitness plot indicates that the model has a high distance error value at generation 10. The increase in the distance scalar's weight causes temporary chaos. The process recovers and quickly proceeds to an optimal solution.

8.3.2 Generational joint freedom

The second speed-up test involves restricting joint motion on a generational basis. The sequence in which joints are given freedom is presented in Table 8.4. While again this approach does not speed up the evolutionary process, it is clear from the average fitness plots in Figure 8.8 that the varying joint freedom grammar produces very stable gaits from the earliest generations.

Table 8.4: Generational joint freedom. Only joints with a ✓are free to move and evolve motion data for each generation range. All other joints remain static. The numerical value ranges in the top line of the table indicate the generations for which the joint freedoms apply.

	0 - 10	**11 - 20**	**21 - 30**	**31 - 50**
Scapula (fore)	✓	✓	✓	✓
Shoulder (fore)	✓	✓	✓	✓
Elbow (fore)	-	✓	✓	✓
Carpal (fore)	-	-	✓	✓
Fetlock (fore)	-	-	-	✓
Hip (hind)	✓	✓	✓	✓
Stifle (hind)	-	✓	✓	✓
Tarsal (hind)	-	-	✓	✓
Fetlock (hind)	-	-	-	✓
Proximal (neck)	-	✓	✓	✓
Mid (neck)	-	-	✓	✓
Atlas (head/neck)	-	-	-	✓

The large starting values apparent in most of the average fitness plots are the result of the unviable phenotypes passed to the simulation application, usually at the start of the evolutionary process. The motion data

can cause the model to wildly gyrate its limbs or provide such a boisterous gait that the model flips over. These bad phenotypes are awarded a very high score (corresponding to the worst fitness possible). By initially restricting the model's degrees of freedom, production of the bad phenotypes appears minimised.

While these experiments were attempting to speed up evolution by varying the environment, in the final experiment of this chapter, the issue of varying terrain is addressed.

8.4 Uneven terrain experiment

Terrain traversal is a very large and open issue in computer animation and especially robotics (see Section 2.4.1).

Uneven terrain often features in animated movies and video games. Poor foot placement is frequently observed and quickly breaks the illusion of reality. Even in some of the most graphically advanced video games currently available, a kinematically animated character's feet are often observed to break contact with the ground and appear to float above, or sink through, uneven surfaces and stairs.

Uneven terrain is a particularly challenging issue for physics-based animal models as it involves balance, foot placement, path planning and in many cases, emulation of visual and tactile feedback. While the work presented in this thesis does not directly address topics of balance and terrain, we have applied the GE-based motion generation system to a very basic terrain traversal problem.

A simple terrain is constructed with an ODE triangular mesh primitive. The physics-based model is then directed to walk over this terrain using motion data which is optimised for walking over a flat surface. While the model does make it over the terrain, a testament to the inherent stability of the motion data, much slippage of the hooves occurs. There is also unwanted impacts of the hooves with the ground surface during the transfer phase.

Screenshots of the model walking over the terrain using the flat surface motion data are shown in Figure 8.9 **A**. In this figure, images taken at three sequential time points are superimposed onto a single image.

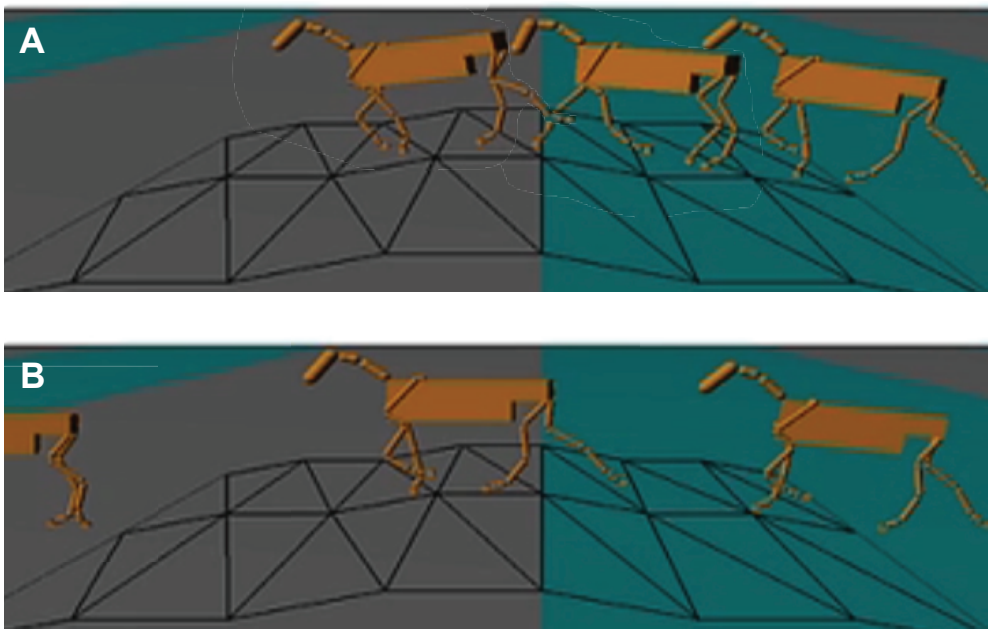


Figure 8.9: The unoptimised **A** and optimised **B** terrain traversal motions are compared by displaying three images taken at the same timepoint as the horse traverses the terrain using both forms of the motion data. It can be clearly seen that the model moving with the optimised motion traverses the terrain more quickly.
Video 8.6 Terrain traversal unoptimised
Video 8.7 Terrain traversal optimised

The GE-based motion optimisation approach described in the previous sections is then applied to the terrain traversal problem. In this approach, the grammar is first altered to produce a phenotype with six sets of motion data cycles, to be used in sequence as the model moves over the terrain. This modified grammar is presented in Listing 8.3. The difference between this grammar and the original one presented in Listing 8.1 is in the top three lines which dictate that six separate cycles of motion data are output separated by the delimiter 'T'. To provide the model with a stable motion to start evolving from, the seed data included in this grammar is previously optimised for the model on a flat surface.

The horse simulation fitness function is then modified to accept this multiple motion pattern data format. The fitness function itself is weighted to reward distance travelled in a certain number of cycles. The distance component, mentioned in the previous section, is included in the fitness function for the terrain optimisation experiment as components such as duty factor are arguably less important in this situation. On flat surfaces, the distance travelled is very closely related to stride length, however, on uneven terrain the stride lengths achieved may vary from cycle to cycle.

The value of the distance component is multiplied by its fitness weight as shown in Equation 8.8. The total fitness function score is then calculated using Equation 8.9.

$$\text{distance score} = \text{distance travelled} \times \text{distance weight} \tag{8.8}$$

```
<prog> ::= <t_curves> <newline> <t> <newline> <t_curves> <newline> <t> ...

<t> ::= T

<t_curves> ::= <fcurve0> <newline> <fcurve1> <newline> ... <fcurve11>

<fcurve0> ::= <curve0> | <curve0> + <funcs>
...
<fcurve11> ::= <curve11> | <curve11> + <funcs>

<funcs> ::= <funcs> <op> <funcs>
          | <function>
          | <med_freq_amp_var>

<op> ::= + | -

<function> ::= <low_freq_amp_var> * sin( <low_freq_var> * 2 * PI * t )
             | <low_freq_amp_var> * cos( <low_freq_var> * 2 * PI * t )
             | <med_freq_amp_var> * sin( <med_freq_var> * 2 * PI * t )
             | <med_freq_amp_var> * cos( <med_freq_var> * 2 * PI * t )
             | <hi_freq_amp_var> * sin( <hi_freq_var> * 2 * PI * t )
             | <hi_freq_amp_var> * cos( <hi_freq_var> * 2 * PI * t )

<low_freq_var> ::= 1 | 2
<med_freq_var> ::= 3 | 4
<hi_freq_var> ::= 5 | 6 | 7 | 8

<low_freq_amp_var> ::= 0 | 0.125 | 0.25 | ... | 10
<med_freq_amp_var> ::= 0 | 0.05 | 0.1 | ... | 2
<hi_freq_amp_var> ::= 0 | 0.025 | 0.05 | ... | 0.5

<curve0> ::= 3.63+9.04*sin(1*2*PI*t+-1.04)+2.43*sin(2*2*PI*t+3.42) ...
...
<curve11> ::= ...
```

Listing 8.3: An example of a grammar based on the concatenation of sinusoidal functions to the seed data modified for use in a terrain traversal problem. Six cycles of motion data are optimised in comparison to the usual one. (Omitted terms are represented by '**...**').

$$\text{fitness score} = \text{stride score} + \text{duty score} + \text{energy score} + \text{distance score} \tag{8.9}$$

For this experiment, the distance weight is set to be three times that of the other components, e.g. stride weight = 1; duty weight = 1; energy

weight = 1; distance weight = 3. To prevent the evolution from abandoning realistic motion and simply pursuing maximum distance by any means possible, the range of the amplitude constants in the grammar is limited to keep the evolved motions close to the seed data.

The GE optimisation process uses the parameters from Table 8.1. Throughout the evolutionary process, the terrain remains the same and only the contents of the six sequential motion data files are evolved.

8.4.1 Results

Out of the large number of optimisations attempted, the overwhelming majority of solutions traversed the terrain in significantly shorter times than the flat terrain motion data. Images of a model moving with terrain-optimised motion data are shown in Figure 8.9 **B**. This sequence of images was taken at the same time points as the model in **A**, and it demonstrates that the optimised data moves the model at a significantly faster speed over the terrain.

An analysis of the generated motions show that the increase in speed is due to the higher stepping action adopted by the model. By taking higher steps, the unwanted collisions with the terrain during the limbs' transfer phase are mostly avoided and instances of hoof slippage are significantly reduced. An illustration of this high stepping behaviour is shown in Figure 8.10.

In these frames the model can be clearly seen taking a high, extended step up onto the first part of the terrain. By taking this big step, the model avoids accidentally colliding its hooves with the elevated surface and puts the model in a stable position to move over the rest of the

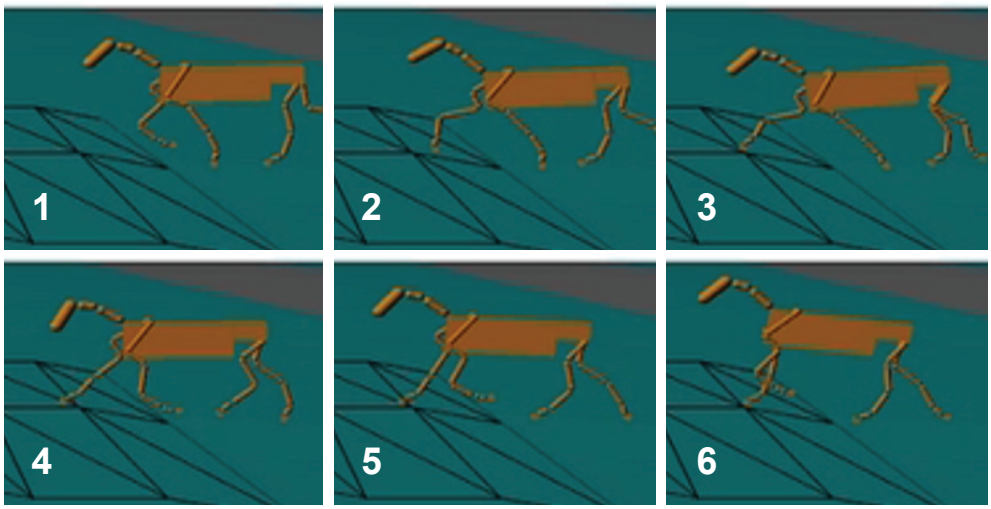


Figure 8.10: Six sequential, numbered screenshots of the physics-based horse model moving over uneven terrain are shown. The motion data optimised for this terrain produces an exaggerated stepping motion as the model moves up on the terrain segment.

terrain in a similar manner.

This is a very positive result, however, the use of a single terrain for both the optimisation and testing are indicative of the limits of this approach. A more exciting contribution would be a GE system that could evolve a controller which dynamically calculates the motion data required to traverse any given terrain.

The evolution of dynamic controllers is not directly explored in this thesis, however, a similar concept is discussed in Chapter 10 when dynamic gait adjusters are described.

8.5 Chapter summary

In this chapter, a GE-based motion data optimisation system for a physics-based horse model is presented.

The optimisation system is a combination of a Java implementation of GE called GEVA, a quadruped-simulation fitness function and a grammar which includes motion data in the summation of sinusoids representation.

The quadruped-simulation fitness function is described in detail referring to its use of dynamic similarity predictions and energy efficiency calculations. Experiments are presented that compare motion data generation approaches that utilise seed motion data to those that don't.

A second group of experiments examine the affects that the dynamic alteration of the grammar and fitness function have on the rate of evolution. The final section of the chapter describes the application of the optimisation system to an uneven terrain traversal problem.

In the following chapter, the motion data optimisation and generation approach is applied to models of different size and the issue of automatic spring-damper coefficient generation is explored.

Chapter 9

Variable morphologies

The subject of motion data reuse for models of different size and proportion is addressed in this chapter.

Given the expense involved in acquiring motion data that pertains to a single real-life animal, when animations of multiple animals are required, that expense is multiplied. It would therefore be valuable to have a system that can reuse a single piece of motion data as the seed for generating motion for other animals of different shapes, size and species.

The first section of this chapter describes an attempt to reuse data measured from an animal of one species to animate a model of another. This motion data retargeting system exploits natural evolution observations to iteratively retarget motion data through a series of hybrid animal models and is presented in Section 9.1.

During this experiment, a significant obstacle to the retargeting approach became apparent; models with varying mass and proportions require different spring-damper (s-d) coefficients. This implies that both the motion data and s-d coefficients must be optimised for every model.

To explore this issue, a method is presented in which data pertaining to a particular horse's gait motion can be used to animate horses of different age, sex, breed and conformation. To demonstrate this method, an application is described in Section 9.2 that automatically generates horse models of a user-specified age, for which motion data can be generated.

Using this application as a testbed for automatic s-d coefficient optimisation, two GE approaches to this problem are compared. In one approach, the model's s-d coefficients are optimised prior to the motion data optimisation. This method is contrasted with a parallel optimisation of both the s-d coefficients and the motion data.

In advance of this, in the following section the motion data retargeting attempt is described followed by a discussion of the issues raised by this experiment.

9.1 Interspecies motion retargeting

The horse is a well-studied animal and as such, there is motion data available in the biology and veterinary literature. With the vast majority of animals however, there is no motion data available. In response to this, the possibility of using data measured from one species to animate another is investigated. This process is referred to as motion data retargeting and the system and experiments presented in this section are published [132].

The concept of motion retargeting is not uncommon in the animation field, however, it is usually employed in relation to human motion capture data [71, 159]. The process of adapting and adjusting motion capture

data for different characters is fraught with difficulty and thus animators refer to it as the "motion retargeting problem". To overcome this problem, several systems are proposed including those that use intermediate skeletons to aid the retargeting process [128].

While these retargeting systems often utilise optimisation processes to some degree, they are frequently based around animation constraints. Our motion data retargeting system in contrast does not operate through constraints, but rather allows a motion to gradually evolve towards the new model.

The evolutionary approach is inspired by the fact that an animal's skeletal dimensions and musculature have gradually adapted to its environment over millions of years through natural evolution [62]. Because of this, the difference in body proportions between two animals of differing species can be large, however, cursorial quadrupeds have highly similar skeletal structures and gait patterns, which potentially allows for motion data retargeting.

Two specific retargeting systems are presented in this section. Both systems take motion data measured from a horse and optimise it for use with a dog model. In one system, a model's motion and the model's bone proportions themselves evolve towards the target animal in one continuous optimisation process. In the other system described next, a number of separate optimisation processes are carried out on a predetermined number of hybrid models.

9.1.1 Discrete model system

The discrete motion data retargeting system is shown in Figure 9.1.

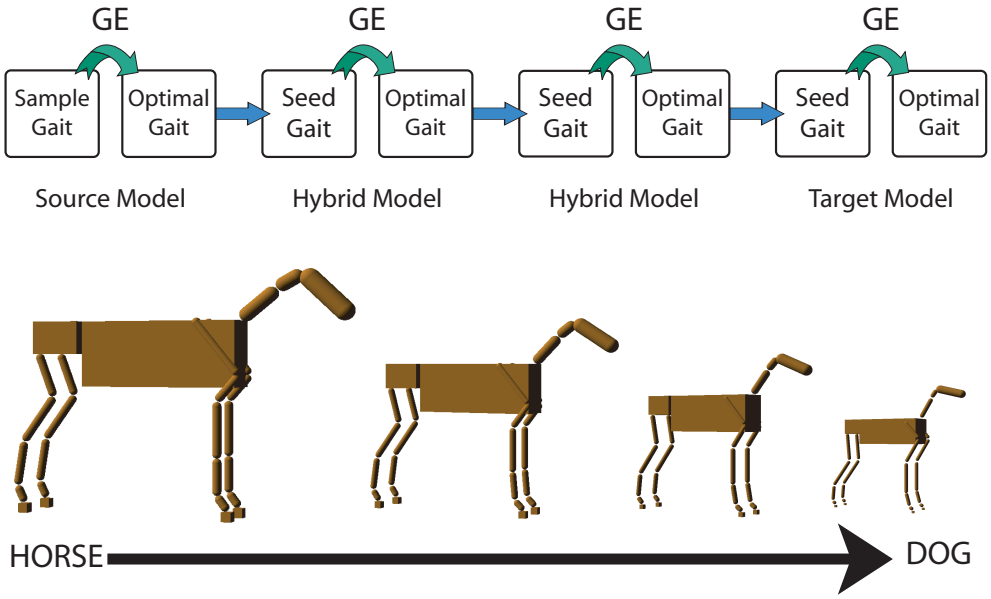


Figure 9.1: Discrete motion data retargeting with Grammatical Evolution (GE). Optimisations are performed on a series of intermediate horse-dog hybrid models.

This figure shows the standard physics-based horse model on the left and a dog model on the right constructed from published skeletal data [12]. For each motion data retargeting attempt, the system constructs a user-specified number of hybrid models, each of which is a linear interpolation between the horse and dog model. As the gait retargeting process proceeds, the bone proportions of subsequent hybrid models tend towards those of the target animal.

The retargeting system utilises the motion data generation method and application described in Chapter 8. The motion data is represented in the summation of sinusoids format and optimised using the combined numerical and concatenating functions approach.

The fitness function is again based in part on dynamic similarity theory which is highly suitable as its predictions should hold true for all cursorial quadrupeds travelling at the same Froude number. The other component of the fitness function is energy efficiency which is also theoretically an appropriate choice. Each animal's musculoskeletal system has evolved to minimise energy expenditure and the consequent differences in musculature and bone proportions between species result in different motion patterns. The theory is that as the incremental optimisation process progresses, each newly generated motion pattern will further resemble the target model's real-life motion, due in part to the energy efficiency-based scoring system.

The retargeting process starts by generating an optimal motion data cycle for a pure horse model from the measured seed data. This optimised motion data then becomes the seed data for a hybrid model, for which the GE system generates a new optimal motion data cycle. This process is repeated until a pure dog model is reached and its motion optimised. This is a simplified overview but further details of the retargeting process are given in following subsection.

Process

The discrete motion data retargeting process utilises a combination of applications, data files, control scripts and grammars. Both the GEVA system and Physics-based Quadruped Simulation (PQS) application presented in Chapter 8 are utilised in the retargeting process, as is an additional application called the Grammar Writer (GW).

The GW is a simple application that takes both a grammar format file and a set of numerical range parameters as input. Using the grammar format and value range data, the GW outputs a BNF form grammar that can be directly used with a GE system. The GW is used throughout the retargeting process to take the optimised motion data from one hybrid model, and incorporate it into a new grammar as seed data for the next hybrid model in the sequence.

As the retargeting process involves optimising motion data for models of differing mass and proportions, varying s-d coefficients are required. The retargeting system uses an automatically generated set of s-d coefficient values which each are a function of an input parameter. For each hybrid model, a large range of input parameters are tested and the best performing value is selected and used for the motion data optimisation.

To assess the suitability of a set of s-d coefficients, an additional component called motion curve deviation is added to the fitness function. This component scores how closely a set of s-d coefficients can keep the movement of the bones synchronised with the target motion data.

During each cycle of a run, the difference between the actual angle and target angle of each bone in the model is recorded. For each limb an average disparity value for the entire run is calculated and the values are summed. This sum total is then normalised to a range appropriate to the other fitness components and is finally multiplied by its fitness weight.

This measurement of how appropriate the generated joint torques are, coupled with the gait quality fitness score, provides a usable indication of how suitable a set of s-d coefficients are for a particular model. The subject of s-d coefficient optimisation will be revisited in Section 9.2.

Prior to the start of the retargeting process, the user specifies the number of the hybrid models to use, the Froude number and gait pattern. During the optimisation process, the limb phase differences of the gait pattern are imposed on all hybrid and target model simulations via the motion controllers. The degree to which the generated motion data may deviate from the seed data is also set by the user through arguments passed to the GW. Certain data files must also be supplied including the source and target model files and the initial seed motion data.

The main sequence of operations that occur during the gait retargeting process are presented in the following list. Data movement and other low-level operations are omitted for conciseness.

1. Retargeting process commences
2. Simulation application calculates and outputs hybrid model files
3. GW takes the current best seed motion data and outputs a grammar file to GEVA
4. Simulation application evaluates a range of s-d coefficient input parameters, storing the best one
5. GEVA optimises motion data for the current model using the supplied grammar file and simulation application as fitness function
6. Steps 3 to 5 are repeated for each hybrid model until either a solution is found or a certain number of generations is reached

Discussion

The retargeting system operates excellently for the hybrid models that are close in size and proportion to the original horse model. Issues arise

with the s-d coefficients however, as the process progresses further towards the target model.

The best chosen generic set of s-d coefficients score worse with each subsequent hybrid model. It appears that as the mass of the model decreases, the sensitivity to poorly set s-d coefficients increases. Although the retargeting process does complete in most cases, the retargeted motion is ultimately evolved to compensate for poorly set s-d coefficients. The resultant motion is recognisable as a natural gait but contains many erratic gestures.

It appears however, that the discrete hybrid model system would perform better if the s-d coefficients could be set correctly for each model. A numerical optimisation approach to s-d coefficient generation was tested but it too experienced problems.

If s-d coefficients are to be optimised for a model, motion data that allows the model to move in a stable manner must be available. Unfortunately, that stable motion data cannot be generated without an appropriate set of s-d coefficients.

In response to this problem, a continuous retargeting approach is presented in the following section.

9.1.2 Continuous model system

Rather than evolving motion for a set of distinct hybrid models, the continuous system evolves the model itself towards the target.

This approach is illustrated in Figure 9.2. The retargeting system attempts to optimise the motion data, s-d coefficients and the model itself towards the target at the same time. The grammar is written in

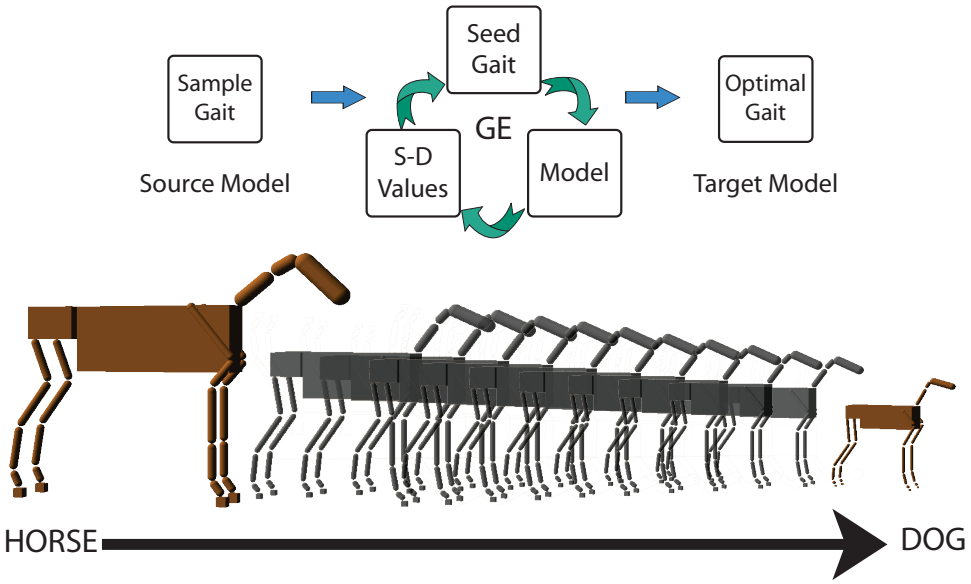


Figure 9.2: Continuous motion data retargeting with Grammatical Evolution (GE). Optimisations are continuously performed on the seed data, model's proportions and s-d coefficients until the target is reached.

such a manner that the phenotypes produced from the GEVA mapping process contain model construction-, s-d coefficient- and motion-data.

This continuous approach necessitates the addition of yet another component to the fitness function. As the model itself is being evolved, a score is given for how much the current model differs from the target, rewarding those models which tend to be smaller and more doglike.

Process

Prior to the commencement of the continuous retargeting process, the Froude number and gait pattern are set by the user and the system is supplied with seed motion data, a source and target model and the set of s-d coefficients that were manually tuned for the horse model (Section 7.2.4).

Each of these source data files are input to the GW application and a grammar is output according to the user-specified parameters which control the optimisation value ranges within the grammar. The numerical parameters of each phenotype are chosen from these ranges.
During the continuous retargeting process the following, simplified sequence of steps are taken.

1. GW takes the current best seed motion data, source and target model files, and manually tuned s-d coefficients and then outputs a grammar file to GEVA
2. GEVA optimises motion data, s-d coefficients and the model towards the target using the supplied grammar file and simulation application as fitness function
3. Step 2 is repeated until either a solution is found or a certain number of generations is reached

Discussion

Attempting to optimise three separate and interdependent components at the same time is a significant challenge as the sheer number of variables involved produces a huge search space.

During experimentation, the system never completed a full retargeting, however, at the end of a set number of generations, the resultant model/motion/s-d coefficient combination appears very stable in a large number of cases. In the majority of the 30 runs carried out, the model evolves towards the target model proportions but then stops at roughly the same point.

One could speculate that this is a tipping point, after which the tolerance of incorrectly set s-d coefficients significantly lowers. Regardless of cause, the biggest problem appears to be that there are too many conflicting objectives in this continuous approach. It is possible that with a more sophisticated fitness function, the evolution could be guided towards the target. It is also possible that attempting to create motion in a physics-based model using this retargeting approach is either unfeasible or computationally intractable.

9.1.3 Retargeting summary

The motion data retargeting experiments presented in this section were undertaken with an appreciation of how difficult a task the retargeting would be.

While stable retargeting was not achieved, the attempts resulted in the development of several interesting evolutionary systems. Each experiment also reminds us how important an appropriately chosen set of s-d coefficient values are to a physics-based animal model of this nature.

In the final continuous experiment, the optimisation of s-d coefficients appeared to work well, as long as the model did not stray too far from the state in which it had appropriate motion data to move with. Motivated by this finding, the issue of automatically generating s-d coefficients for animal models that differ in proportions within a particular species is explored in the following section.

9.2 Multiple models

In this section, the animation of horse models of varying shapes and sizes is examined.

Animations of large herds of animals are often featured in movies, television and video games. If a scene is to be considered realistic by a viewer, the diversity of shape and size of the animated herd's individuals must be as found in nature. As such, many different-shaped animal models must be constructed and animated.

The variation in a herd of animals comes from several factors. Assuming each animal is of the same species, for animals like the horse, it is possible that a herd contains multiple breeds. Within those breeds the animals may vary in age and sex. In addition to this, individual animals exhibit different conformations. The variation in the shape and size of each animal will affect its motion. If an animation of such a herd of animals is to be realistic, motion data that relates to each individual animal is required.

In this section, a system is described for constructing and animating physics-based horse models of different shapes and sizes using construction and motion data pertaining to a single animal. To produce the variety of models required for a herd scene, an application is presented that can automatically generate and construct horse models of a user-specified age. The body proportions of the generated models are determined by user supplied animal allometry data. The model generation system and experiments presented in this section are published [135].

When a model is constructed, motion data is optimised for that model using the same GE-based motion data optimisation technique described in Chapter 8 and used for the motion retargeting experiments. The large range of body shapes and sizes that can be generated is problematic however, as each novel model requires a bespoke set of s-d coefficients. This issue was apparent in the previous section and it was concluded that an automatic approach to s-d coefficient optimisation is required.

In Section 9.2.3, a method by which s-d coefficients can be optimised both prior to and during the motion data optimisation process is explored. In Section 9.2.2 the age specific model construction and motion generation system is described. In advance of this, the allometric data used by this system is presented.

9.2.1 Allometric measurements

In Section 4.3.5, the topic of allometry was introduced. For the experiments presented in this section, observations of skeletal allometry for a horse are utilised in the animation system.

The growth-rate of the body segments delineated in Figure 4.7 are plotted in Figure 9.3 for both colts (male) and fillies (female). The data on which this figure is based is taken from [188] which presents a study of the skeletal growth-rates of Thoroughbred horses.

The original data is measured from over 100 Thoroughbred horses in two week increments from birth to 84 weeks [188]. For each of the body measurements mentioned above, average growth data for both colts and fillies is published. The power law equations shown in Table 9.1 are based on this data and are used to create the plots shown in Figure 9.3.

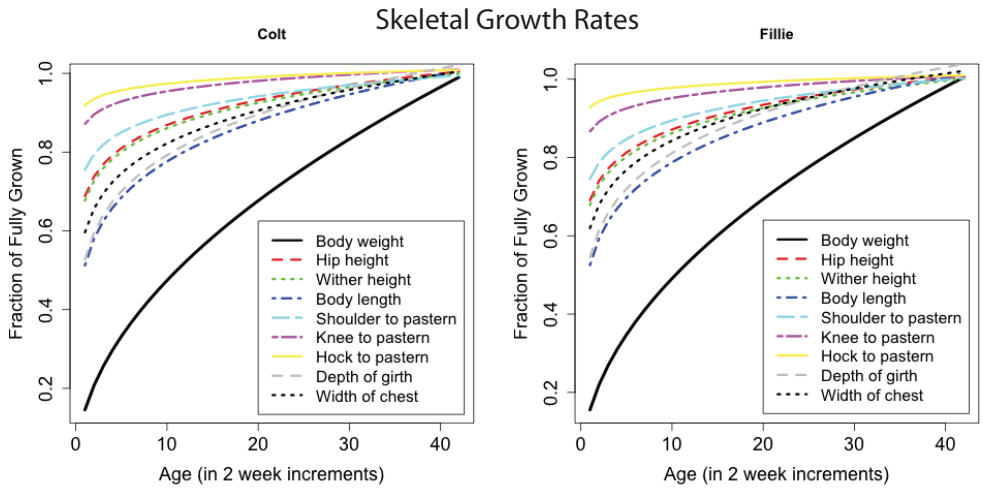


Figure 9.3: Skeletal growth-rates for Thoroughbred horses taken from [188], approximated by the power law equations in Table 9.1.

The power equations presented in Table 9.1 are used to create distinct physics-based horse models in the Variable Morphologies System described next.

9.2.2 Variable Morphologies System overview

The Variable Morphologies System (VMS) automatically generates and constructs a physics-based horse model of a user-specified age. The breed of horse generated is determined by the input model construction file and the supplied growth-rate data which defines how different aspects of an animal's body grow in relation to each other as the animal ages.

The model values presented in Figure 9.4 *(right)* are calculated using the filly growth-rate equations in Table 9.1 which are also plotted on the left of the figure. Each model's characteristic values are a fraction of the base horse model data which is the standard data used for each of the horse models in this thesis, provided in Appendix Section B.1.3.

Table 9.1: Skeletal growth-rate power law equations.

Body segment	Equation (Colt)	Equation (Fillie)
Body weight	$y = 0.1454 * x^{0.5132}$	$y = 0.1555 * x^{0.4992}$
Hip-height	$y = 0.6882 * x^{0.1012}$	$y = 0.6907 * x^{0.1009}$
Wither height	$y = 0.6767 * x^{0.1044}$	$y = 0.6785 * x^{0.104}$
Body length	$y = 0.5121 * x^{0.1806}$	$y = 0.5252 * x^{0.1758}$
Shoulder to pastern	$y = 0.7553 * x^{0.0736}$	$y = 0.7452 * x^{0.0793}$
Knee to pastern	$y = 0.8717 * x^{0.0394}$	$y = 0.8667 * x^{0.0407}$
Hock to pastern	$y = 0.9194 * x^{0.0249}$	$y = 0.9288 * x^{0.0224}$
Depth of girth	$y = 0.5269 * x^{0.1771}$	$y = 0.5453 * x^{0.1725}$
Width of chest	$y = 0.5962 * x^{0.1395}$	$y = 0.6197 * x^{0.1336}$

The VMS is an extension of the quadruped simulation application (PQS) described in Chapter 8. The modified system takes the base model data and calculates a new model's construction data from the growth-rate equations, according to a user-specified age. This new model is then constructed in the standard manner described in Section 5.3.1.

It should be noted that there is an age and breed mismatch between the growth-rate equation horses and the horses from which the base model data comes from. This is because published studies regarding growth-rates and anatomical data for specific breeds are often unavailable, however, the incongruity between these two sets of data is acceptable for proof of concept.

Depending on data availability, the VMS system can be easily modified to generate models varying in breed, conformation, sex and deformity. Regardless of how a model is generated however, motion data must be optimised for use with that model. The following section describes

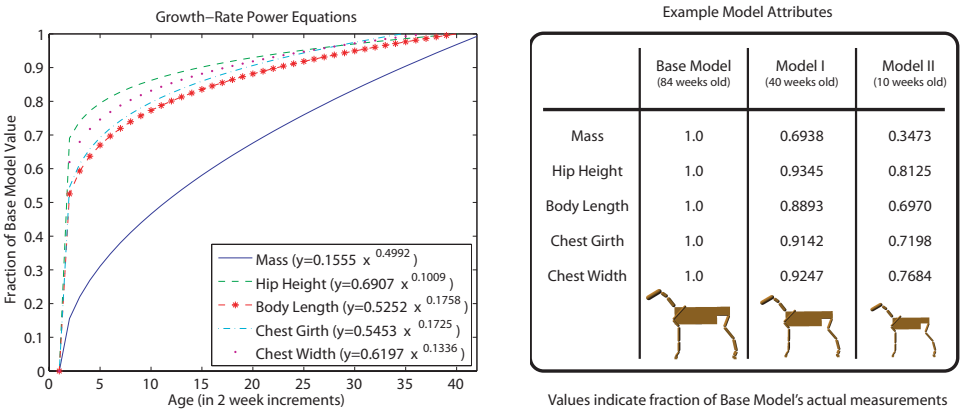


Figure 9.4: Growth-rate equations *(left)* and example model values *(right)*. The example models shown are fillies in this case.

the motion data generation process and further discusses the problem of s-d coefficient tuning.

Motion data optimisation process

The variable morphology motion generation approach uses the same GEVA-based system described in Chapter 8 except that the VMS acts as the fitness function.

The optimisation process uses the combined numerical and concatenating sinusoidal functions grammar. This approach ensures that the optimisation remains constrained to the realism of the seed data, whilst providing scope to deviate, thus accomodating different morphologies.

Once a candidate phenotype is generated from this grammar it is passed to the VMS for evaluation. The model, for which the gait is being optimised, is constructed and moved for a few gait cycles using the generated phenotypic motion data. The movement is then scored by a fitness function and that score is passed back to the GEVA system.

The components of the fitness function score are again based on energy efficiency, stride length and duty factor. As previously mentioned, s-d coefficients must be evolved for each model, as will be discussed in the following section. As with all optimisations of s-d coefficients, the fitness function also includes the motion curve deviation component, as introduced in Section 9.1.1.

9.2.3 Spring-damper coefficient experiments

As stated previously, the tuning of the s-d coefficients is of vital importance to the motion generation process. For a system that dynamically constructs and animates models of different size and proportions, an automatic approach is especially important.

Ideally, exact values for the s-d coefficients could be calculated based upon model mass, model hip-height, bone length, bone mass and position in the hierarchy. Attempts were made to use GE to evolve equations, using the aforementioned model traits, which would allow one to easily calculate the s-d coefficients for any model. All efforts to produce an equation for the entire model, and each joint respectively, were wholly unsuccessful, most probably due to the very large number of variables involved.

As this ideal approach remains elusive and possibly unachievable, a numerical optimisation approach is employed instead. From the failed experiment mentioned above, some valuable knowledge was gained and is used to constrain the numerical optimisation.

The general range of s-d values which will allow a model to move are linked to the mass of that model, however, the model's skeletal structure

is also influential. Bones higher up in a limb have a much greater mass to move and therefore require greater torques. The bones lower in a limb have less mass to move, however, some bones are subject to large angular velocities as the model moves and may have to generate compensatory torques in order to achieve their own target motion.

In addition to this, the previously stated s-d coefficient setting problems still stand. Setting the spring constant too low will not allow the model to apply the necessary torques to move according to the motion data. Setting the spring constant too high can cause the simulation to explode and crash. This risk of an invalid simulation run due to a single extreme value prevents us from running a numerical optimisation on an unconstrained range of values.

Optimisation system

The s-d coefficient optimisation system is a two-step process. The system firstly runs a rapid series of tests to ascertain the range of s-d coefficient values that will provide the most stable gait cycle possible given motion data that is not optimised for that particular model. The numerical optimisation is then performed within that range. The entire optimisation process is illustrated in Figure 9.5 and is explained in detail below.

Prior to commencement, an s-d base range, sufficient for base model motion, is experimentally established. As the base model pertains to fully-grown horses, all generated models will be smaller in mass and dimensions. It is therefore assumed that each model's ideal s-d coefficient range will fall beneath this maximum value.

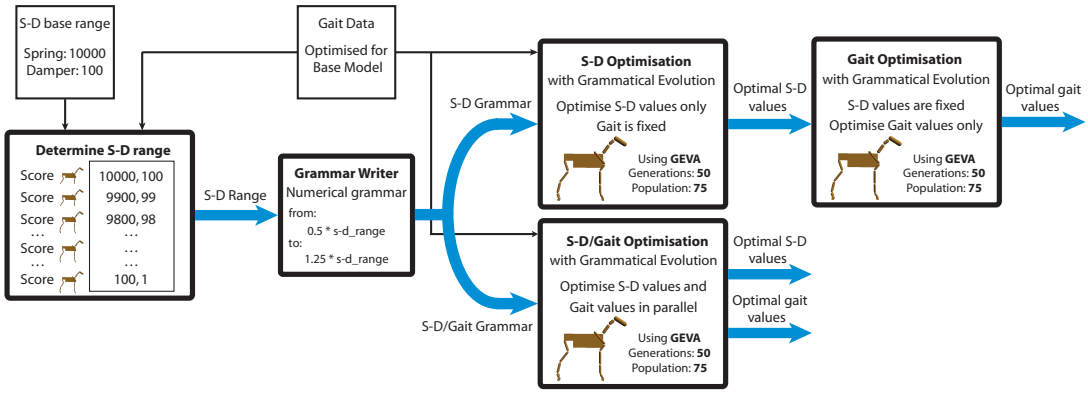


Figure 9.5: Flow chart of the spring-damper (s-d) coefficient and gait optimisation process. Note: the two distinct approaches investigated are distinguished by the fork in the flow chart.

The process begins with a rapid test to determine what s-d value range is appropriate for the generated model. The motion data used to test each value was previously optimised for the base model. This motion data should not produce perfectly stable motion as the generated models differ in proportion, however, the gait cycle should be sufficiently stable to provide an indication of performance. At this point, the VMS fitness function is weighted to score the motion data based only on the motion curve deviation component.

The generated model is then automatically tested with a set resolution of increments of the s-d base range values, with each bone in the model given the same s-d coefficient parameters. For a more stable gait, each joint will require its own specific coefficients but at this stage, the uniform values tested simply indicate an acceptable range. The values which give the most stable gait throughout the testing dictate the range of values used for the s-d coefficient numerical optimisation, and are passed into the Grammar Writer application.

Each grammar is automatically written to allow for a numerical optimisation within a set range. In these experiments, the range allowed by the grammar spans from 0.5 up to 1.25 times the passed-in values. This range is divided into 50 even increments and this set of values is written into the grammar. Once the numerical optimisation range is established, there are two separate approaches to the optimisation, as detailed in the following subsections.

In the first approach, indicated by the upper path after the fork in Figure 9.5, a numerical optimisation is performed to establish the specific s-d coefficients for the entire model. The seed motion data is then optimised for use with the model. The second experiment shown on the lower path, optimises the s-d coefficients and motion data in parallel.

Each of the experiments presented in this section are performed using GEVA with the parameters presented in Table 8.1. For each approach, results are presented for two generated models, corresponding to models I and II as shown in Figure 9.4.

Sequential spring-damper and motion data optimisation

In this first experiment, the s-d coefficients are optimised separately to the motion data. The results of 30 runs are presented in Figure 9.6. As with all the experiments presented in this thesis, a lower fitness score indicates a better result. The plots show large variance in fitness across all of the runs with the variance most obvious for model II.

It can also be seen that in the earliest generations, both the best and average fitness scores are higher for model II (and therefore worse) than for model I. This may be because model II is a younger horse. Being

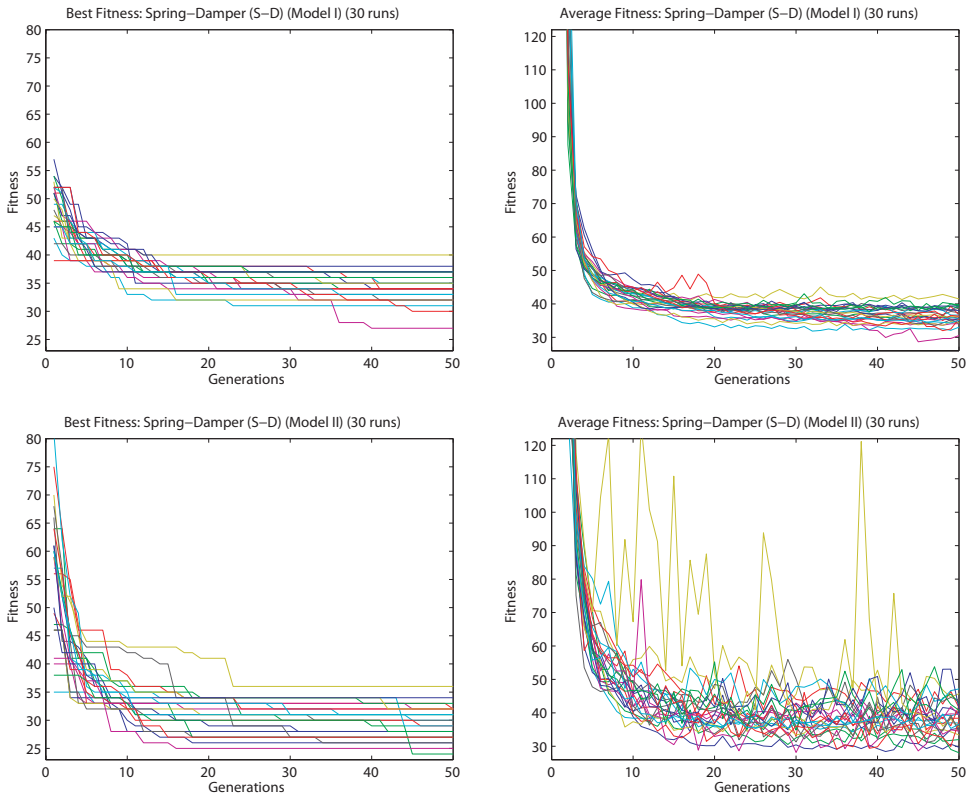


Figure 9.6: Best fitness and average fitness for s-d coefficient optimisation (models I and II, 30 runs).

younger, it is therefore more different in body proportion to the base model than the older model I. The motion data cycle used for these s-d coefficient optimisations is optimised for the base model and it has been observed that the more a model differs from this base, the less stable the gait. Instability yields bad fitness scores, and this may explain these results.

The average of the best and average fitness scores is shown in Figure 9.7. It is surprising that the best fitness score of model II is significantly better than model I despite it starting off worse. Model II may be scoring highly because the s-d coefficient optimisation is compensating for the unstable gait through its evolution of the s-d values. Although interest-

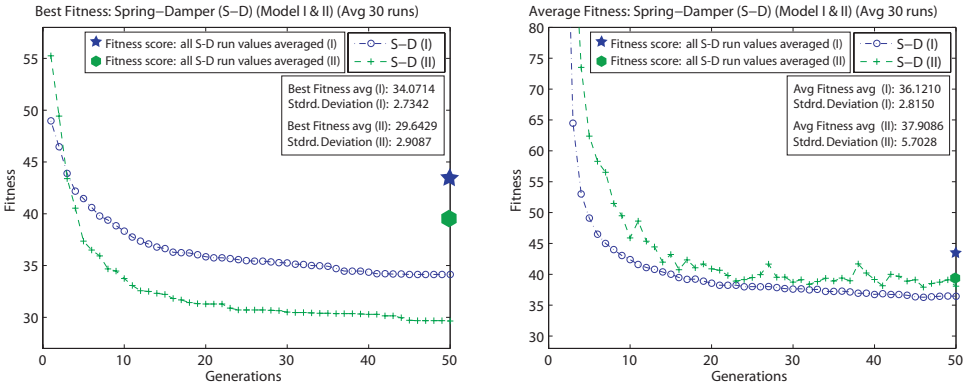


Figure 9.7: Best fitness and average fitness for s-d coefficient optimisation (models I and II, averaged 30 runs). Standard deviation values are included for the averaged best and average fitness scores. The generated s-d values for all 30 runs were averaged and scored for comparison.

ing, this is undesirable as we wish to evolve the best values for future motion data optimisations, rather than values that provide stable motion from unsuitable motion data.

It can be seen from the average fitness plots in Figure 9.7 that model II's average fitness score was worst overall. Model II's average fitness standard deviation is also significantly higher than model I's. This high disparity in average score between runs again indicates the high level of instability caused by the base model's inappropriate motion data.

During this experiment, is was also hypothesised that the average of all the generated s-d coefficients would yield a high-scoring set of values. The average of all the s-d value sets, from each of the 30 runs, is therefore calculated and experimentally tested in the simulation application; these scores are indicated in Figure 9.7. The hypothesis appears to be incorrect as the scores achieved are significantly worse than the best fitness scores of the non-averaged individual runs. This perhaps demonstrates that the performance of the values in each generated set are dependent on

one another and that there are a huge number of potential solutions for each model's s-d coefficient optimisation.

Once a suitable set of s-d coefficients are found for a generated model, the motion data optimisation process can take place. This is a completely separate optimisation process again using GEVA with the parameters presented in Table 8.1. In total the sequential optimisation process runs over 100 generations; 50 generations for the s-d coefficient optimisation and then another 50 generations for the motion data optimisation.

The results of these runs will be discussed after the parallel s-d/motion data optimisation approach is briefly described.

Parallel spring-damper coefficient and motion data optimisation

In this second experiment, the s-d coefficient values are optimised in parallel with the motion data.

In this system, once the best performing s-d coefficient range is established, the Grammar Writer creates the grammar in the manner described previously in this section. This time however, the grammar also includes the motion data optimisation terms, similar to the continuous retargeting grammar described in Section 9.1.2.

The value ranges used in all aspects of this experiment are the same as the sequential approach, but instead of having two sequential runs of 50 generations, the optimisation is initially performed in a single run of 50 generations. The results are presented in Figure 9.8.

9.2.4 Comparison

A comparison between the sequential and parallel gait optimisation approaches is shown in Figure 9.8.

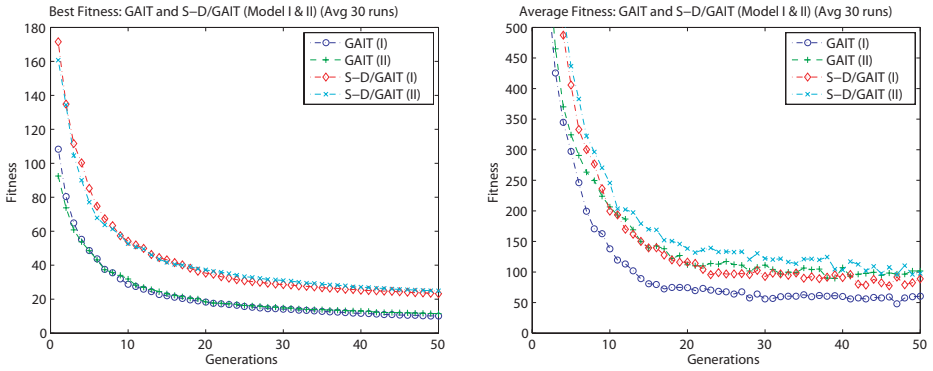


Figure 9.8: Best fitness and average fitness for sequential gait and parallel spring-damper (s-d) coefficient and gait optimisation (models I and II, averaged 30 runs).

It can be seen in Figure 9.8 that the sequential optimisation approach to optimisation has won in terms of best fitness score. The average fitness scores show model II scoring worse than model I in both experiments, regardless of optimisation approach. It should be noted that the sequential approach had double the computation time of the parallel approach, however, further experiments in which the parallel approach was given equivalent generations did not show a significant improvement.

Overall the sequential approach performs well, however, the fact that the s-d coefficient optimisation is performed with motion data which may not be stable for a diverse range of models is a potential problem. This problem might have been resolved by using the parallel optimisation approach, however, this has performed poorly, perhaps due to the large number of variables in the optimisation.

In future, the parallel approach could be improved by imposing certain constraints during the optimisation in an attempt to reduce the number of free variables. The best solution may be to try a combination of both sequential and parallel approaches, perhaps using both approaches interchangeably during the optimisation process.

In conclusion, we have successfully developed an application that can automatically produce distinct horse models according to a user's specifications. Using the sequential s-d coefficient and motion data optimisation technique, it is then possible to optimise a single piece of motion data for use with a variety of different models. The animation results of these experiments are discussed in the following section.

9.2.5 Animation results

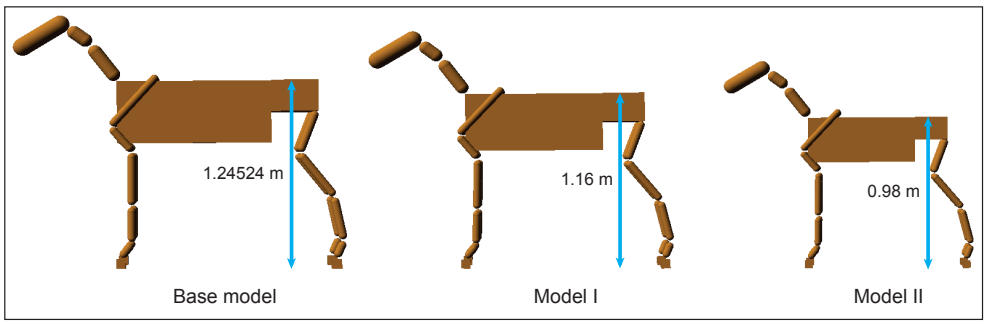


Figure 9.9: The two horse models (I and II) constructed by the Variable Morphologies System (VMS) are shown in the centre and on the right of the image. The horse on the left is the base horse model used throughout this thesis and whose data the other models are calculated from. The displayed hip-heights are measured from the models and closely match those predicted by the allometry equations.

As stated in the previous section, the results of the s-d coefficient optimisations experiments would ideally be improved, however, the best

performing sets of s-d coefficients are perfectly usable for animating animals of varying shapes and sizes.

The VMS accurately calculates and constructs animal models of the correct size and proportion for a given age. An example of this is shown in Figure 9.9. The figure shows images of the base model, used throughout the physics-based experiments in this thesis, and model's I and II, used for the experiments presented in this section.

When the proportions of each of these models are measured, the results closely match the proportions that are predicted for a horse of that age using the allometric power law equations presented in Table 9.1.

Using s-d coefficients and motion data generated using the sequential approach, the generated motions appear appropriate to each model's proportions. Each motion is also distinctive. Sequential frames from a scene in which models I and II are trotting side-by-side are shown in Figure 9.10.

By way of comparison, an identical scene was tested with two models of different sizes, but with the same proportions. As both models were scale replicas of one another, the same optimised motion data was used for each. The homogeneity of both the models and the gait motion was immediately obvious as the animation was visually appraised.

In conclusion, based on a subjective visual evaluation of the resultant animations, it would appear that incorporating knowledge of animal allometry into a herd scene can improve the realism of the animation.

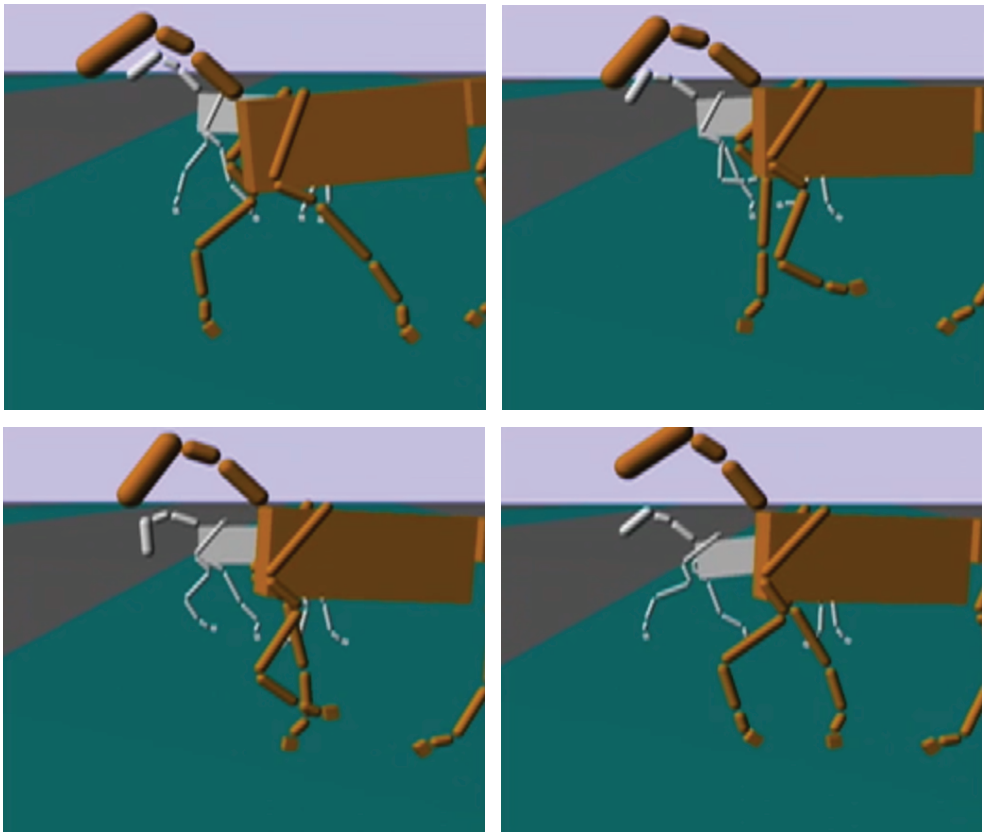


Figure 9.10: Four screenshots from a scene involving two horses of different ages. The brown horse to the forefront of the scene is Model I and the white horse is Model II.
Video 9.1 Multiple models scene

9.3 Chapter summary

In this chapter, the reuse of motion data measured from a single animal, for the animation of other animals of differing age and species is explored.

The first half of the chapter presents an interspecies motion data retargeting system. Discrete and continuous evolution systems are described in detail, including additions to the quadruped simulation fitness function. The retargeting process is found to be unsuccessful and motivates the need for an automatic s-d coefficient optimisation system.

In the second half of the chapter, the automatic optimisation of s-d coefficients is investigated using the Variable Morphologies System (VMS) which generates structurally distinct models based on that model's age. Through the presented experiments, a sequential approach to s-d coefficient optimisation and motion data generation is found to be better than a parallel approach.

Unlike the retargeting process, the VMS generated models and motions are stable, realistic and can be used to add an extra element of realism and visual interest to an animation involving multiple animals. The difficulties involved in producing physics-based animal animations are still apparent however, as a large amount of manual development and computational expense is involved in producing a single piece of stable motion data for a single model.

To investigate the possibility of enforcing realism on an animation, without the associated problems of a physics-based system, the issue of using realistic gaits and transitions for kinematic animal animations is explored in the final chapter of Part III.

Chapter 10

Kinematic gait and transition animation system

In this chapter, the issue of realistic animal animations is examined in relation to kinematic models.

A common problem with animal animation is when the animal models move with gait patterns that are inappropriate to their velocity. For example, a model translating at a slow speed whilst its limbs move in a gallop pattern is visually incorrect and physically implausible. Similarly, a model translating at a fast rate whilst using the limb pattern of a walk is equally incorrect. The animation system presented in this chapter, and published [133], addresses this problem.

If an animation is to be realistic, the model should move with a gait appropriate to its velocity and be able to transition smoothly between gaits when necessary. As such, this chapter focuses on the production of a range of motions for each of the natural gaits and the transitions between those gaits.

The experiments presented in the previous chapters of Part III have been based upon physics-based animal models. While the animations resulting from this type of simulation can be highly realistic, the motion data generation and animation process is nontrivial and computationally expensive. Additionally, each of the presented experiments focus on optimising a single gait motion for a particular model.

In contrast to this, the animation system presented in this chapter adopts a kinematic animation approach to avoid the issues of instability and long simulation times associated with physics-based models. By sacrificing some of the physical realism of a physics-based simulation, a larger variety of motion can be quickly generated and used in a real-time animation system.

One of the major contributions of this animation system is the method by which it dynamically alters a gait's motion data depending on the model's Froude number. These dynamic gait adjusters are described in Section 10.2 along with a description of how they are evolved using GE and an extension to the Curve Modifier Application (CMA) introduced in Chapter 7. Following this, the animation system and transition calculations are each described in Section 10.3.

One of the novel aspects of this system is its hardware controller which allows a user to control a model's Froude number in real-time. The operation of this controller is discussed in Section 10.3.2.

The final section of this chapter presents experimental data measured from the animation system and discusses the resultant animations. In advance of this, a brief overview of the animation system is provided in the following section.

10.1 Animation system overview

The animation system presented in this chapter, referred to as the Kinematic Gait Transition System (KGTS), kinematically animates a horse model to move with gait patterns appropriate to its velocity and transition between adjacent gaits when necessary.

The horse model is constructed as a hierarchical kinematic model as described in Section 5.2.2. The user controls the model's motion in real-time through use of a novel animation control system which utilises a Musical Instrument Digital Interface (MIDI) controller to manipulate the model's Froude number.

The Froude number determines the gait pattern with which the model moves based on dynamic similarity predictions. These predictions also dictate when a model should transition to another gait. When this occurs, the transition patterns are dynamically calculated by the KGTS, as will be discussed in Section 10.3.1.

In addition to moving with the correct gait, for increased realism, intra-gait increases in velocity cause an increase in limb extent as well as stride frequency. To dynamically adjust motion data for the current velocity, gait adjusters are evolved using a GE-based system and an extended version of the CMA introduced in Chapter 7. These gait adjusters are the subject of the following section.

10.2 Gait adjusters

The goal of the KGTS is to enable the model to move realistically at any Froude number, utilising the appropriate gait pattern. To avoid

generating and storing separate bone rotation data for every possible Froude number the model may use, a gait adjuster system is employed.

In this system, each of the four natural gaits has its own motion data file in the summation of sinusoids representation and a specific gait adjuster function file which can modify its corresponding motion data depending on the current Froude number.

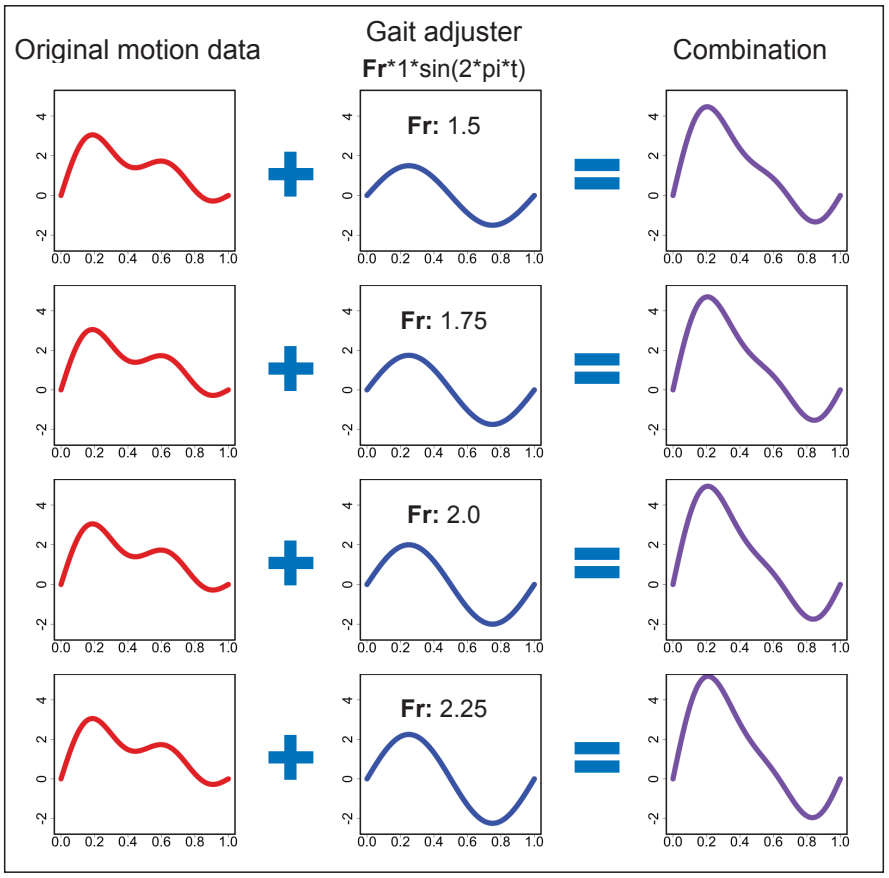


Figure 10.1: Illustration of the gait adjuster system. Each adjuster is a function of the current Froude number **Fr** and is added to the current gait's motion data to produce the motion with which the model is animated.

This motion adjuster function exploits the adjustability of the sinusoidal representation as illustrated in Figure 10.1. Each gait's adjuster

file contains a summation of sinusoidal functions of one or more frequencies whose amplitudes are a function of the current Froude number. The example in the figure shows a single term gait adjuster function, however, an adjuster function can be composed of as many terms as necessary. Each gait adjuster file contains a gait adjuster function for each movable bone in the model.

A well-defined gait adjuster will modify the current gait's motion data to produced realistic motion whilst exhibiting the correct limb extent and duty factor for the current Froude number. Production of such a gait adjuster is complex and as such, the Curve Modifier Application (CMA) introduced in Chapter 7 is extended to aid the process.

10.2.1 Extensions to the CMA

The visualisations provided by the CMA can help a user to manually develop gait adjuster files which perform satisfactorily for a range of Froude numbers.

In addition to the CMA's features described in Section 7.1, the application is extended specifically for the gait adjuster development process. The CMA can be set to automatically test an input gait adjuster file for a specified range of Froude numbers and return a score based on dynamic similarity predictions and limb extent. In the example video listed below, a gait adjuster file causes the red bone and motion curve to change corresponding to the current displayed Froude number.

Video 10.1 Curve Modifier Application

As the CMA tests a gait adjuster file for each Froude number, that Froude number is substituted for the free variable in the gait adjuster data, depicted as **Fr** in Figure 10.1. The user can then view the effect of the gait adjuster on both the motion curves and the limb motion in tandem through the CMA's visualisations. Changes can be manually made to the gait adjuster files, and the result of these changes can then be visually evaluated.

A user manually developing a gait adjuster file must assess whether or not a gait adjuster produces realistic motion that adheres to the musculoskeletal constraints of the real-life animal. In addition to the visual aspect of the gait adjuster development process, the CMA also quantifies the quality of each file as a gait adjuster is scored on its performance at each Froude number in terms of duty factor and limb extent. The user must therefore ensure that a gait adjuster file also conforms to these predictions.

CMA scoring system

The duty factor score is calculated using dynamic similarity predictions. As the animation is not physics-based, measuring the duty factor is not simply a case of recording when a hoof is in contact with the ground. Instead, the position of the hoof is measured throughout the gait cycle. These positions are compared to the position of the ground plane, correcting for total limb rotation. If the hoof is within a specified threshold distance of the ground plane vector, that limb is considered to be in its stance phase.

Unlike stride length, limb extent cannot be directly predicted using dynamic similarity. Limb extent is measured as the distance between the farthest hoof position (in contact with the ground) attained in the forward and backward directions over a single cycle. In a real-life animal, limb extent depends on the animal's velocity and gait pattern, with greater limb extent usually associated with greater velocity.

For each gait, a range of limb extent values are estimated. The absolute maximum limb extent value is determined by how far the model can physically extend its limbs. The minimum extent is based on data pertaining to collected walks [36]. Within these maximum and minimum limits each gait is assigned a limb extension range. The slower gaits have low minimum and maximum extent ranges and the faster gaits have high minimum and maximum extent ranges. A galloping horse for example, can extend its limbs to the maximum limits but cannot reduce its limb extent to the lower ranges of a walk.

For each gait, the actual limb extent value predicted for a particular Froude number is calculated as a function of that Froude number and the gait's calculated limb extent range.

Even with the assistance of the CMA's scoring system however, manual gait adjuster development is nontrivial and an automated approach is preferable.

10.2.2 GE system

To automate the gait adjuster development process, a GE-based system is used to evolve the gait adjusters.

Using GEVA and a suitable grammar, phenotypes are produced in the gait adjuster data file format. Each file contains a different gait adjuster function for each movable bone in the model. Each gait adjuster function comprises a summation of sinusoids of differing frequencies, whose amplitudes are a function of a free variable and may also be subject to a combination of mathematical operators and constants. This concept is illustrated in the examples shown in Figure 10.2. An example grammar is shown in Listing 10.1.

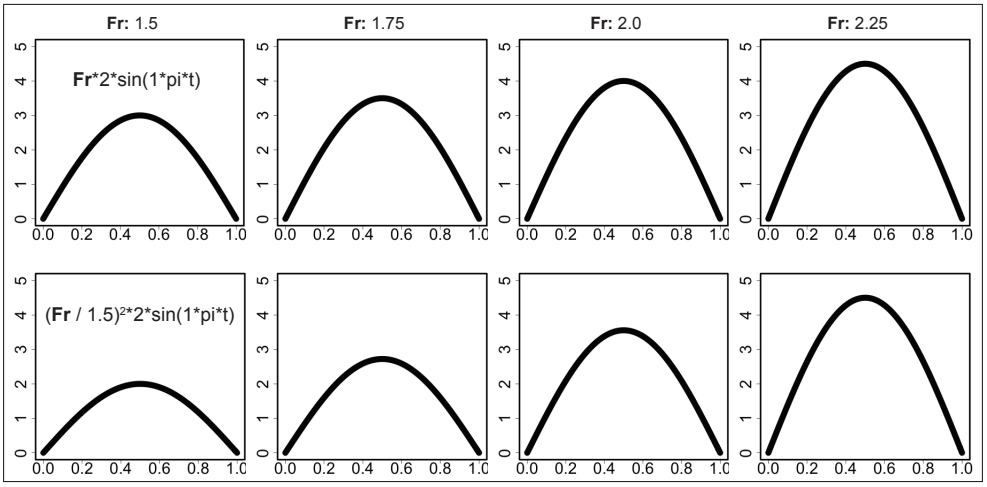


Figure 10.2: Example gait adjuster curves are displayed for a range of Froude numbers **Fr**. The difference in growth-rate of the two curves demonstrates how by combining mathematical operators, free variables and constants, a rich variety of adjuster curves may be evolved.

During the optimisation process, evolved gait adjuster files are passed to the CMA for assessment. The scoring system described in the previous section acts as the fitness function and the visual aspect of the CMA is disabled. Each phenotype is tested for a range of Froude numbers and a fitness score is returned to GEVA. This process continues until a solution is found or a set number of generations is reached.

```
<prog> ::= <fcurve0> <newline> <fcurve1> <newline> ... <fcurve11>

<fcurve0> ::= <funcs>
...
<fcurve11> ::= <funcs>

<funcs> ::= <funcs> <op> <funcs>
          | <function>
          | <Fr> * <function>
          | <zero>

<op> ::= + | -

<zero> ::= 0

<Fr> ::= <Fr> <fr_op> <Fr>
        | Fr
        | <low_freq_amp_var>

<fr_op> ::= + | - | / | *

<function> ::= <low_freq_amp_var> * sin( <low_freq_var> * 2 * PI * t )
             | <med_freq_amp_var> * sin( <med_freq_var> * 2 * PI * t )
             | <hi_freq_amp_var> * sin( <hi_freq_var> * 2 * PI * t )

<low_freq_var> ::= 1 | 2
<med_freq_var> ::= 3 | 4
<hi_freq_var> ::= 5 | 6 | 7 | 8

<low_freq_amp_var> ::= 0 | 0.125 | 0.25 | ... | 10
<med_freq_amp_var> ::= 0 | 0.05 | 0.1 | ... | 2
<hi_freq_amp_var> ::= 0 | 0.025 | 0.05 | ... | 0.5
```

Listing 10.1: An example of a gait adjuster grammar. (Omitted terms are represented by '**...**').

Using a grammar similar to the example in Listing 10.1, a satisfactory gait adjuster file is produced rapidly. The best and average fitness plots of the gait adjuster evolution are shown in Figure 10.3. The kinematic nature of the fitness function means that the evolution proceeds at a much greater pace than a physics-based system. The kinematic model also avoids issues with stability of motion and therefore the gait adjusters can simply approximate the desired motion without adverse consequence.

Once gait adjuster data files are evolved for each of the natural gaits, they are supplied to the KGTS, whose operation is described next.

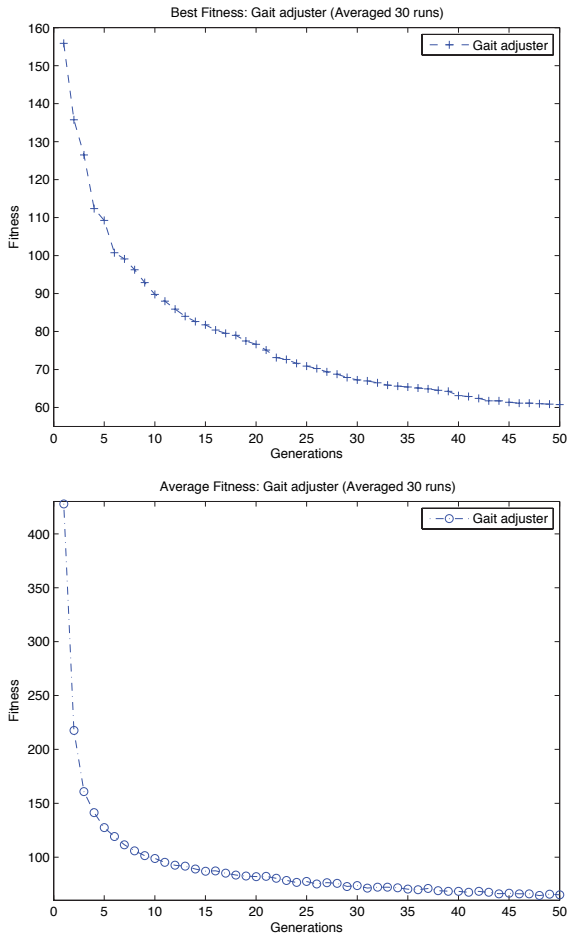


Figure 10.3: Gait adjuster evolution. Best and average fitness (averaged, 30 runs) are presented on the top and bottom respectively. *(Note difference in scale.)*

10.3 Kinematic Gait Transition System

In this section the functioning of the KGTS is described. The KGTS animates a horse model using gaits and transitions determined by that model's Froude number. As such, information on model construction, gait patterns, transitions and dynamic similarity must be provided.

The KGTS is supplied with a model data file (Appendix Section B.1.3) and from this, the model is constructed and animated as a hi-

erarchical kinematic model, as described in Section 5.2. An additional file containing 3D polygon mesh data for skinning the model can also be provided (see Section 2.4.1).

The KGTS must also be supplied with gait phase data (Table B.3), transition Froude values (Table 4.1) and dynamic similarity power law equation values (Table 4.2). A motion data file, originally optimised for use with the physics-based horse model is also supplied for each of the gaits, in the sinusoidal representation. Finally, a gait adjuster file which is generated using the GE-based approach described in the previous section is provided for each of the natural gaits.

The system begins by reading the model and motion data files. Before the animation starts, the velocity at which the model should move is calculated based on the current user-specified Froude number and the model's hip-height. From this Froude number, the current gait is also determined and the gait adjusters are then calculated and added to the current gait's original motion data. This data is then converted to a set of rotation values for each bone.

The final value to be calculated is the stride frequency, using Equation 4.4. The limb extent exhibited within a particular gait depends on the current Froude number and the stride frequency must be modified to compensate for how the limb extent affects velocity. This final stride length value is then used to control the rate at which the animation cycles through each bone's rotation data and thus the velocity of the model when the animation begins to run.

During the animation, the rotation values are cycled through producing motion until a change in Froude number is flagged. If this new

Froude value is within the range of the current gait, the gait adjusters are recalculated to modify the limb extent for the next cycle. If the Froude value is outside of the current gait's range, a transition is flagged.

10.3.1 Transitions

When a transition is flagged, the system identifies the two gaits involved in the transition. For each bone in the model, the final rotation point of the current gait cycle is stored. The gait that is being transitioned to is then created. The gait adjusters are calculated and added to the original motion data for that Froude range and the first rotation point of that motion data is stored.

The transitional data is then calculated as the average of the two sinusoidal waveforms (with zero offset) it will be transitioning between. This average waveform is then converted to the piecewise representation (Section 6.2.2). The start and end points of the transition data are adjusted to join up with the stored offset values, as shown previously in Figure 6.8.

The relative timing of the horse's footfalls vary from gait to gait, as was discussed in Section 4.4.2. The timing of each limb's motion is referred to here as that limb's phase value; each value describes the timing of a footfall with respect to the start of the gait cycle. It should be noted that this concept of phase is completely unrelated to the phase values of the sinusoids which describe a joint's motion.

During a transition, each limb adjusts itself to its new phase value by either increasing or decreasing its rate of movement. It achieves this by spreading its motion data over a number of gait cycle stages different to

the standard number of stages in a cycle (100). To simplify the system, each limb's transition is completed over a single transition cycle in which that limb's motion is either extended, reduced or unchanged.

For each limb the phase difference value for the current gait is subtracted from the phase difference value for the gait that is being transitioned to. This value will be referred to as the transition phase number, or **TP**. If the absolute value of **TP** is less than 50, than **TP** is added to the standard number of stages in a cycle, i.e. 100, yielding the number of transition cycle stages. If the absolute value of **TP** is greater than or equal to 50, the absolute value of **TP** is subtracted from 100. This resultant value is then subtracted again from 100 if **TP** is positive and added to 100 if **TP** is negative, yielding the number of stages in the transition cycle.

Table 10.1: Calculation of the transition cycles stages required for a canter (current) to a gallop (target) transition. (**TP**: transition phase number.)

Limb	Current	Target	**TP**	Calculation	Transition
Fore-left	0	0	0	100 + 0	100
Fore-right	80	10	-(70)	(100 - 70) + 100	130
Hind-left	80	50	-(30)	100 + -30	70
Hind-right	50	60	10	100 + 10	110

An example of this calculation process is shown in Table 10.1. The value on the far right of the table is the number of transition cycle stages and, if the value is not equal to 100, the transition motion is spread across this extended or reduced cycle, thus changing the phase difference of the limb.

The preceding calculations ensure a transition is never excessively long or short relative to the standard number of cycle stages, creating smooth transitions between gaits. As previously mentioned, these transitions are caused by changes in Froude value, as described next.

10.3.2 MIDI control

During the animation, the user can control the model's motion by adjusting its Froude number. Given the dynamic nature of the system, a complementary control interface has been developed for it. A Korg nanoKONTROL slim-line MIDI controller [104] is used to adjust the model's Froude number and initiate transitions in real-time.

The animation system is controlled by five vertical sliders and a toggle button associated with each slider. An additional Transition button flags a transition (when in the manual transition mode). A simplified illustration of the controller is shown in Figure 10.4. The user may either manually move through gait cycles or use the automatic transition mode in which the system automatically chooses a gait to transition to, based on the current Froude value.

In manual mode, to intiate a Transition the user selects a gait adjacent to the current gait, sets the slider and presses Transition. Figure 10.4 presents an example situation. Assuming the current gait is a Walk with a Froude value of 1.05, Trot is selected and its Froude slider set to 1.8. The Transition button is pressed and the model transitions from Walk to Trot during the following cycle. The system allows for continuous up and down transitions between adjacent gaits.

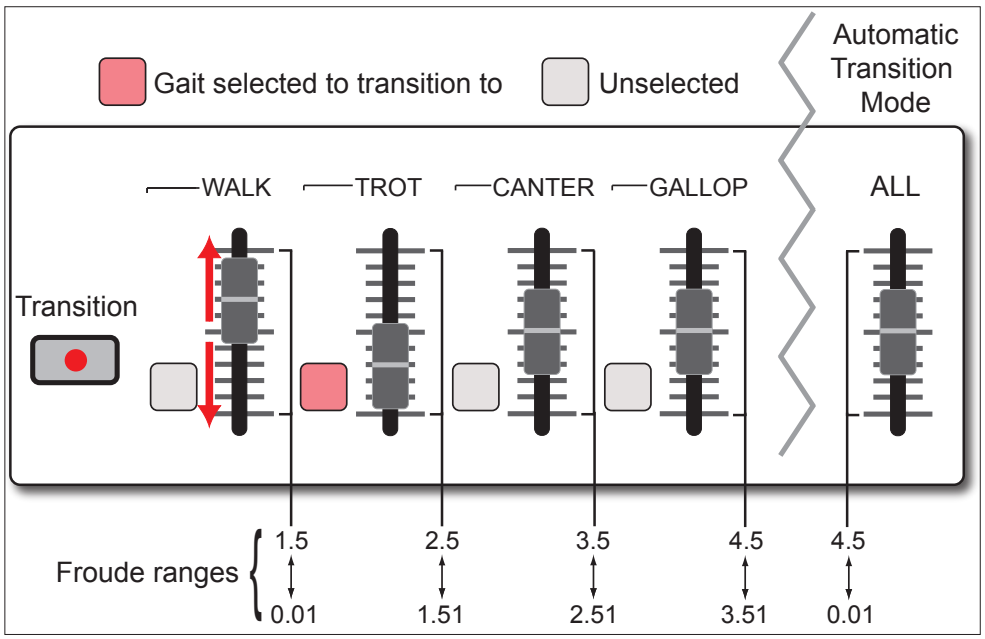


Figure 10.4: Simplified illustration of the MIDI controller. A user selects a gait for transition, sets the slider to a Froude value and then presses the Transition button.

The automatic transition mode gives a single slider control over the animal's Froude value. The user sets this slider to some value along the entire Froude range. Depending on a setup option, the application either checks for a Froude value change at the start of each cycle or upon a press of the Transition button. If the Froude value has not changed, the model continues to move as it was, otherwise the application checks if the change indicates a transition. The model will then transition if required.

If it is flagged as a simple change of Froude within the current gait, the gait adjusters will be calculated and applied. To simplify the system, any requests for transitions between non-adjacent gaits are ignored.

The Froude number can also be controlled through an input plan file, written prior to the animation's commencement. A user can specify an indefinite sequence of Froude number changes and the number of

cycles between Froude changes. The experimental data presented in the following section is collected through this planned approach.

10.4 Animation results

In this section, a series of experimental measurements and animation data are presented and discussed.

The most important outcome of the KGTS is the resultant animation and the quality of the gaits and transitions it displays. In the example video listed below, the MIDI control system is demonstrated with the horse model transitioning up and down through the four natural gaits.

Video 10.2 Kinematic Gait Transition System

As a visual assessment of any animation is subjective, the calculated transition phase data and the effect of the gait adjusters on the bone rotation data are also analysed.

The data presented in this section is collected from a single animation run using the sequence of gaits and transitions shown in Table 10.2.

Table 10.2: KGTS animation plan.

Gait	Froude number	Gait cycles
Walk	1.0	3
Trot	2.0	3
Canter	3.0	3
Gallop	4.0	3
Canter	3.0	3
Trot	2.0	3
Walk	1.0	3

10.4.1 Transition phase data

In Figure 10.5, the limb phase data from the test run is presented. The diagram displays the phase difference between limbs (thick black bars represent ground contact) during a continuous run starting from the top-left. The data is divided into standard gait cycles by the vertical bars and is therefore not displayed temporally and does not indicate velocity.

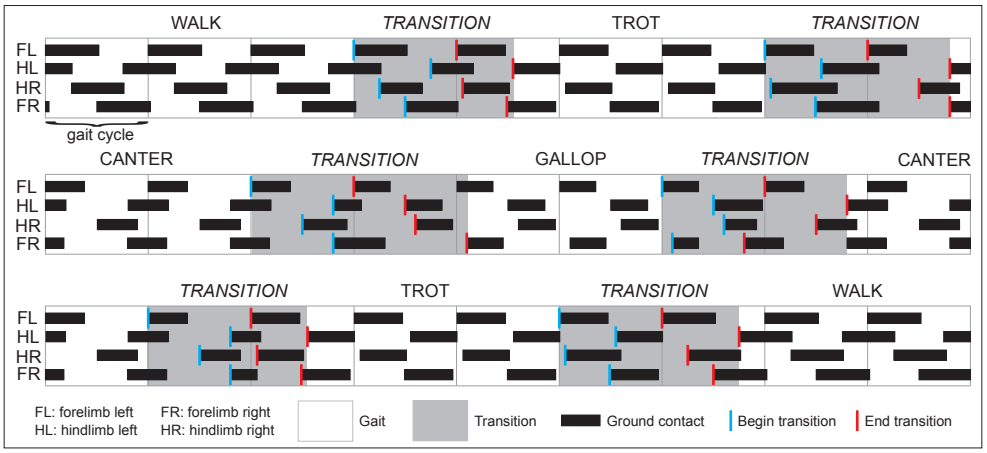


Figure 10.5: Phase differences for gaits and transitions as calculated by the application.

From this diagram it can be noted that each of the gaits has the appropriate limb phase relationship as presented in Table B.3. The gaits recorded through the sequence of upward transitions are identical to the gaits recorded during the downward sequence. The diagram also displays a clear difference between the symmetrical and asymmetrical gaits.

The calculated transition cycle stage values for each limb are presented in Table 10.3. These values are calculated by the KGTS in the manner described in Section 10.3.1. In each transition, the system has ensured that the increase or decrease in movement rate is not extreme compared to the surrounding data.

Table 10.3: Calculated transitional cycle stages.

Transition	Fore-left	Hind-left	Hind-right	Fore-right
Walk to Trot	100	80	81	99
Trot to Canter	100	125	144	131
Canter to Gallop	100	70	110	130
Gallop to Canter	100	130	90	70
Canter to Trot	100	75	56	69
Trot to Walk	100	120	119	101

10.4.2 Bone rotation data

The plot of a bone's rotation during the same test run is presented in Figure 10.6. The stretching effect of the transition cycle can be seen in the upper image as can the effect of the gait adjuster. For this example, identical gait data is supplied for the four natural gaits. The evolved gait adjusters dynamically adjust the data to provide greater limb extent in the faster gaits and this is reflected in the curve amplitudes.

The lower image displays rotations with respect to a time unit (length of a Walk cycle). As the model transitions up through the gaits, the stride frequency increases. This can be seen to the right of the lower image as the gait cycle motion curves occur more frequently per time unit.

The effect of the gait adjuster on the motion curve's shape is quite obvious in Figure 10.6. The smaller peak present in the motion curve at lower Froude numbers gradually disappears as the Froude number increases for the faster gaits. In this case, the smoothing of the motion curve is justified, however, in some cases the gait adjusters may smooth over important nuances of a motion.

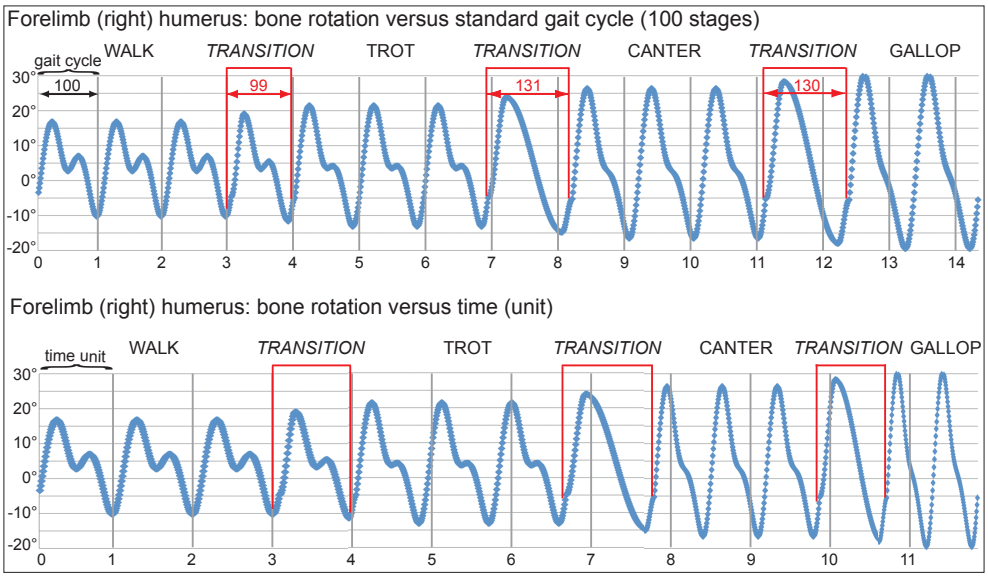


Figure 10.6: Rotation of the right forelimb's humerus during locomotion. The upper image plots rotation against a standard gait cycle. Transition durations are given in gait cycle stages. The lower image plots rotation against a time unit equal to one walk cycle.

The extent to which a gait adjuster alters data can be constrained through the grammar by incorporating knowledge of the original gait motion data. In addition to this, the gait adjusters could benefit from a more sophisticated fitness function during their evolution.

10.4.3 Aesthetic realism

The experimental data presented in the previous sections demonstrate that the KGTS can mimic certain quantifiable aspects of natural motion.

The ultimate reason that detailed gait and transitional information is included in an animation system is realism. If the on-screen motion pattern of each gait and transition does not look realistic, than the system has failed, regardless of what an analysis of the motion data concludes.

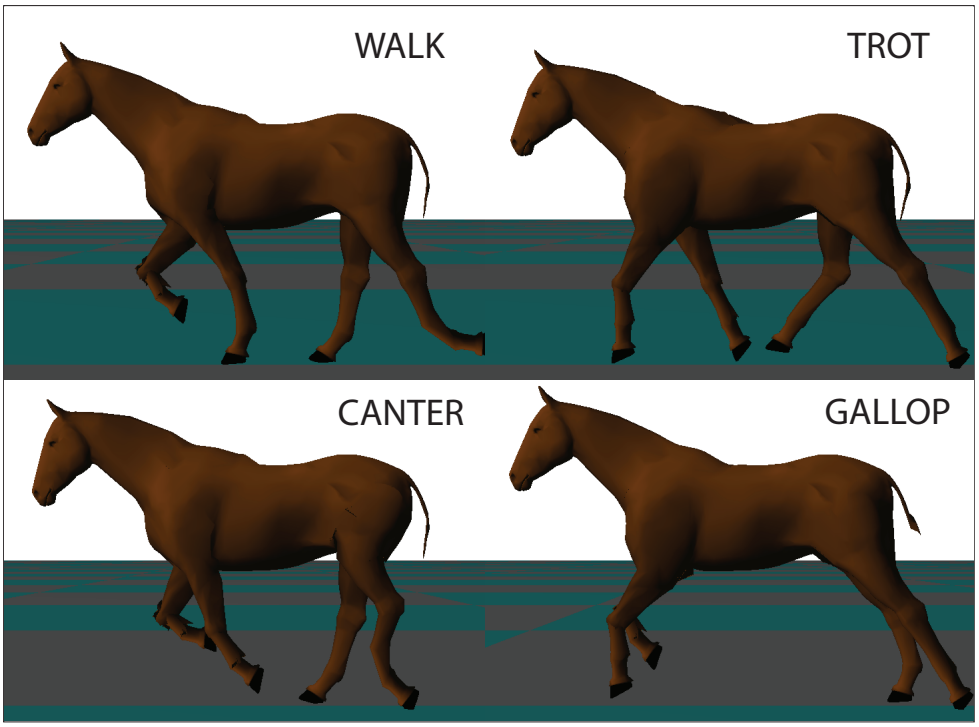


Figure 10.7: The horse model in motion for the four natural gaits.

Figure 10.7 shows screenshots of the four natural gaits taken from a single run. A visual inspection of the animation shows a model moving with a natural looking gait. As the initial planned velocity is relatively slow, the model moves with a walking gait and translates appropriately. The hooves each appear to make contact with the ground and give the illusion of a model thrusting itself forward, despite the kinematic nature of the model.

When a transition is triggered the model smoothly and almost imperceptibly increases or decreases its limb movement to match that of the target gait. The model now translates at a faster pace as it moves with a livelier trot.

The trot to canter transition involves some of the largest phase differences between gaits, yet provides a completely smooth and highly convincing motion. The difference between the symmetrical trot and the asymmetrical canter is immediately apparent. As the limbs move in synchrony, the impression of greater impulsion is given. In the final upward transition, the model seamlessly begins moving with a gallop. At close to maximum speed, the model's limbs are nearly fully extended. The transitions back down through the gaits are equally smooth.

Overall the motion appears convincing as the model moves with gaits appropriate to its velocity and transitions smoothly when necessary.

The most obvious aspect of the animation that could be improved upon however, are the motions of the back. As the model is not physics-based, it does not react to gravity or the forces that would be exerted on it in real-life. As such the back is completely static. A simple solution to this would be to provide vertical back motion data to the KGTS, if such data could be acquired or generated. A more sophisticated solution would involve a real-time analysis of the gait pattern from which the appropriate back motion could be calculated.

Another obvious issue with this animation is a lack of flexibility in the lumbosacral joint. During the faster gaits, a horse will often move its hindquarters in underneath itself to produce greater impulsion. This motion is known as "tucking" and would significantly increase the realism of the animation.

10.5 Chapter summary

In this final chapter of Part III, an animation system is presented in which a kinematic horse model moves with gaits appropriate to its velocity and transitions when necessary.

Gait adjusters are introduced which allow a piece of motion data to be dynamically adjusted according to the model's Froude number. The gait adjusters are generated using a GE-based optimisation system and an extended version of the CMA, originally described in Chapter 7.

The animation system itself is described in terms of its input data and how the transitions are dynamically calculated. A novel MIDI control system that allows a user to control an animal's velocity in real-time is also presented.

The final section of the chapter discusses some data measured from an animation run and an evaluation of the resultant animation is given. It is concluded that accurate modelling of gaits and transitions adds to the realism of an animal animation.

10.6 Part III summary

The objective of Part III was to describe the experiments that were carried out as evolutionary computation techniques are applied to animal animation problems. The experiments presented in this part comprise the major contributions of this thesis. As such, they will be summarised in the final part of this thesis on conclusions and future work.

Part IV

Conclusions

Chapter 11

Conclusions and future work

In the previous chapters, the application of evolutionary computation techniques, specifically GE, to animal animation has been thoroughly explored through research, development and experimentation.

In this final chapter, the presented work is summarised in terms of its contributions and several research questions are answered. This thesis then concludes with Section 11.2, in which potential directions of future research are proposed.

11.1 Thesis summary

In this thesis, we have explored many aspects of animation, natural computing and biology in the search for ways to increase the realism of animal animations using GE. In Part I, extensive background material, some of which can be directly used to improve animation realism, was presented.

In Part II, the origins of both the model construction data and gait motion data were explored. The construction and animation of kinematic

and physics-based horse models using this data was also detailed. In the final chapter of Part II, a manual attempt to optimise motion data for a physics-based horse model using the Motion Data Development Environment was described. It was the difficulties experienced with this manual process that motivated the GE-based automatic motion generation systems and experiments.

Each of these experiments focused on a particular research question posed in Section 1.1.1. In the following section these thesis research questions are restated and the contributions that resulted from their investigation are summarised.

11.1.1 Contributions

In Section 1.1.1 it was stated that the overall goal of this thesis is to explore how observations of natural evolution and evolutionary computation can be used to produce realistic quadrupedal animal animations, specifically focusing on GE.

A set of thesis research questions was also presented in Section 1.1.1. These questions were answered through the development of each of the main contributions of this thesis. The summaries of these major contributions, presented previously in Section 1.2, are reproduced here adjacent to their corresponding research question.

1. *Can GE be used to optimise motion data for a physics-based quadruped model?*

   (a) **Automatic physics-based motion data optimisation**

   A GE-based system for optimising motion data for a physics-based quadruped model is presented in Chapter 8 and is published [134].

   Of all the evolutionary computation methods, Genetic Algorithms (GA) are most commonly applied to motion data optimisation problems [73]. Our aim is to explore the use of Genetic Programming (GP), specifically GE, for the generation of motion data.

   A GP technique's ability to evolve the structure of a solution, rather than optimise a fixed set of parameters as in GA, deem it worthy of investigation for this class of problem. In addition, recent research indicates that GP techniques perform well if not better at certain motion generation problems [175].

   The system presented in this thesis signifies the first time GE has been applied to an animation problem, specifically that of motion generation. In this system a GE implementation controls the evolutionary search while a physics-based quadruped simulation application acts as the fitness function.

   Animating physics-based quadruped animal models is a challenging issue in computer animation. The motion data optimised by GE moves a physics-based quadruped model in a realistic and stable manner.

(b) **Measured motion data**

The issue of the inclusion of domain knowledge into the gait generation approach, in the form of motion data measured from a real-life animal, is also tackled in Chapter 8 and is published [134].

The question of domain knowledge inclusion and optimal AI ratio in GP is open [151]. We compare grammars that optimise motion data and grammars that are free to evolve novel motion. Some of the free-style grammars have no domain knowledge whatsoever and others have knowledge of an animal's muscles implicitly included in the grammar.

Those grammars that optimise motion data produce highly realistic, physics-based animal motions. The free-style grammars in comparison produce unique motions that allow the model to locomote at a high rate of speed, albeit without regard for the physical limits of the real-life animal. In the grammars which implicitly include muscle information, many of the motions produced are comparable to that of a horse. A more sophisticated fitness function may further improve these results.

It is concluded that for realistic motion, domain knowledge must be included to some degree.

(c) **Rate of evolution**

In an evolutionary system, modularly varying goals can speed up the evolutionary process [98]. Inspired by this finding, we

explore how the rate of an evolution can be affected through two experiments, presented in Chapter 8 and published [134].

We investigate whether generationally varying the weights of the fitness function components can replicate this speed-up. In an expansion of this idea, the generational locking and unlocking of joints in the quadruped model is also explored.

Speed-up is only observed locally in the varying fitness function experiment and not at all in the joint locking experiment. In the latter experiment however, the number of valid, stable motion producing phenotypes produced from the earliest generations is greatly increased through the limiting of joint motion at the start of an evolution.

(d) **Terrain**

In the final experiment presented in Chapter 8, terrain traversal is briefly explored. The manner in which a model traverses uneven terrain is another difficult and open issue for physics-based animal models. In response to this problem we employ a GE-based system to generate the motion required to traverse a simple terrain.

The result is impressive as an optimised motion allows the horse model to traverse a terrain at approximately twice the speed of a model whose motion is optimised for a flat running surface. This result demonstrates GE's potential to evolve more sophisticated terrain motion controllers for physics-based animal models and even robots.

2. *Can GE be used to retarget motion data from one physics-based quadruped model to another model of a different species?*

Interspecies motion retargeting

A GE-based system for retargeting motion data measured from one animal to an animal model of another species is presented in Section 9.1 and published [132].

The retargeting of motions between characters often requires much adaptation and adjustment. This "motion retargeting problem", as it is dubbed, remains a topic of much research [71, 128, 83].

The retargeting system presented in this thesis represents the first time an evolutionary computation technique has been applied to a quadruped model motion data retargeting problem. The system uses a GE implementation and quadruped simulation application to evolve motion data through a series of hybrid models towards a target animal model.

Both a discrete and continuous evolution system are described and experiments involving a horse to dog retargeting are presented. The retargeting attempts are found to be unsatisfactory mainly due to the large number of variables involved and inaccuracies of the joint-torque calculations in the hybrid models.

3. *Can GE be used to optimise motion data for multiple physics-based quadruped models of differing skeletal proportions for the animation of herd scenes?*

   Multiple models

   A GE-based system for generating motion data for multiple automatically generated quadruped models of differing skeletal proportions is presented in Section 9.2 and published [135].

   Using a GE implementation and a quadruped simulation application which generates models from allometric data and acts as a fitness function, physics-based horse models of differing proportions are generated based on an age parameter and animated using evolved motion data.

   Sequential and parallel approaches to model parameter and motion data optimisation are also explored. The sequential approach to model parameter optimisation is found to be best and GE is successful at generating motion data for multiple differing models in a herd scene.

4. *Can GE be used to evolve functions that can dynamically adjust a kinematic model's motion based on its velocity and observations of natural locomotion?*

   Kinematic gait and transition animation system

   Research is ongoing on the use of GP techniques for the evolution of controllers for various problems [2, 64]. A GE-based system for evolving functions which dynamically alter a model's motion pat-

tern based on its velocity is presented in Chapter 10 and published [133].

This is the first occasion that GE has been employed for the generation of dynamic motion controllers. These motion adjuster functions are evolved for use in a kinematic quadruped animation system which animates a model with gaits and transitions determined by a user-controlled velocity parameter.

The GE-evolved dynamic adjusters are found to increase the realism of an animation by allowing a model's limb extents to reflect those observed in nature.

11.1.2 Research and implementation decisions

In addition to the thesis research questions posed in the previous section, an additional set of broader questions relating to animation, biology and natural computing were also originally raised in Section 1.3. It was suggested in that section that as the specific thesis research questions were addressed through research, development and experimentation, it would be necessary to answer these broader questions to provide a suitable problem domain in which to explore evolutionary computation techniques and limit the overall scope of the research.

That suggestion was extremely apt, as each of these questions has been answered to some degree, either directly or indirectly over the course of the research. In this section, those questions are stated again and a very brief answer is given to each, based specifically on the findings of this thesis.

Animation

Which animation methods can be used for animal animation?

Both kinematic and physics-based animal animation systems are experimented with in this thesis. Whilst the physics-based animal animations are highly realistic given suitable motion data, the issues with stability and cost of motion data generation can limit their use. Kinematic animations on the other hand are more easily created yet their lack of physical constraints can be obvious unless care is taken when specifying a model's motion.

It can be concluded that physics-based techniques are an excellent choice when highly realistic motion is required, the necessary data is available and issues related to the real-time nature of physical simulation are not a factor. Kinematic animations in contrast are suitable for all situations when extreme physical realism is not a priority. In video games for example, the low computational expense, predictability and simplicity of a kinematic animation often make it the method of choice.

How can animal models be represented, constructed and animated?

The construction of both kinematic and physics-based models is thoroughly discussed in Chapter 5, using skeletal data taken from the biology literature. A simple hierarchical model approach to construction and animation is used for the kinematic model.

The physics-based model is constructed from ODE's simple primitives, using a specific sequence of positioning steps. The model is then animated using motion controllers based on a spring-damper system. The

animation process of both models requires some measured motion data if realism is the goal.

What is the best source of motion data for animations?

It is concluded in Section 6.1 that motion data extracted from photographs is inaccurate and motion capture techniques are prohibitively expensive. The motion data optimisation experiments presented in this thesis all produced motion based on a single piece of data extracted from a published source.

How can motion data be intuitively represented for animation systems?

Rather than store the extracted motion data as a set of discrete values, two intuitive motion data representations are presented in Section 6.2.1 and Section 6.2.2. The sinusoidal and piecewise representations are successfully used in the experiments in this thesis for representing cyclical and acyclical data respectively. The sinusoidal representation in particular is easily incorporated into a grammar and evolved through the concatenation of functions.

Can motion data be manually generated or optimised?

Efforts to manually tune motion data for a particular physics-based horse model were described during a discussion of the Motion Data Development Environment in Section 7.2. While the resultant animation is relatively impressive, the time taken to manually adjust the model parameters and motion data for a single gait renders the manual approach ineffectual.

Biology

Which aspects of an animal's musculoskeletal system are most important when creating an animation?

The horse models used in the experiments presented in this thesis are constructed and animated using accurate anatomical and physiological data. The musculoskeletal system of a horse is explored in Section 4.3 and the simplifications that can be made to the skeleton to produce a model for animation are presented in Section 5.1.2.

It is concluded that the long bones of the skeleton are the most important for modelling motion, whilst the flat bones give a model its structure. Any set of bones connected by joints that allow little or no movement can be treated as a single segment. The physical limits of joints are also important when creating a model. By preventing a joint from rotating past its real-life limit, physical realism is enforced on the model.

In this thesis, the individual muscles of the horse are not considered. Instead the motion that these muscles creates in a bone is modelled through the application of torques about a joint. An animal's muscles could be modelled however, in a more sophisticated model.

What aspects of dynamic similarity theory can be exploited for animation purposes?

The predictions of dynamic similarity theory play a crucial role in every one of the animation systems and experiments presented in this thesis. Described in Section 4.5, a model's Froude number determines the gait pattern with which it moves and the fitness functions used in each

of the motion generation systems rely on dynamic similarity predictions of stride length and duty factor.

Natural computing

How can motion data be represented in a GE grammar?

The process by which motion data, in the summation of sinusoids representation, is included in a grammar is discussed in Section 8.1.1. As well as motion data, evolvable model data and spring-damper coefficients have also been easily incorporated into a grammar in the same manner.

Using an application such as the Grammar Writer, introduced in Section 9.1.1, a user can automatically create a grammar which includes large sets of numerical constants, based on a relatively small set of application parameters.

What type of fitness function can be used to evaluate an animal model's motion?

The fitness functions used in each experiment have differed in terms of their components, yet all have utilised dynamic similarity predictions to some degree. In essence, the fitness function can take any form as long as it can evaluate the phenotypic motion data and return a score based on some motion quality factors. In this thesis, the quality of a gait motion is evaluated in terms of stride length, duty factor and energy efficiency.

Are multivariable fitness functions viable for motion generation problems?

All of the fitness functions presented in this thesis are multivariable to some degree and have performed well in most scenarios. In cases when the evolutionary search space is very large however, the multivariable fitness function is unable to guide the evolution towards its target.

The weighting system of the fitness functions, described in Section 8.1.2, allows a user to tune the fitness function for a particular task. This is important as often two measures of fitness may be in conflict with each other, and the user can decide which component holds more influence in the final score.

How does the size of the search space affect the quality of generated motion data?

For experiments such as the continuous motion retargeting system in Section 9.1.2, the large search space seriously hinders the evolutionary search and the nature of the solutions found can vary hugely from run to run. Based on observations of the experiments in this thesis, high quality motion data is only generated when the optimisation is suitably constrained and the search space reduced.

Would human involvement through an interactive fitness function assist a motion generation problem?

In situations where the optimisation is relatively unconstrained, such as with the free-style grammars in Section 8.2.2, the basic "shape" of the produced motion may be determined in the early generations of the

evolution. The direction taken may exhibit exactly the type of motion one would wish to evolve or it may be essentially unusable. By having a human direct the evolution in the early stages, these less-constrained grammars may become more useful.

Could use of a multi-objective evolutionary algorithm be of benefit for motion data optimisation?

As each of the fitness functions in this thesis are multivariable, a multi-objective evolutionary algorithm may be advantageous [46]. The performance of multi-objective evolutionary algorithms does not scale well with a large number of objectives however, and the experiments in this thesis include up to five fitness objectives.

Our simple weighted multivariable fitness function performs well with the GE search when the optimisation is suitably constrained and it may therefore be more benefitial to focus on developing these constraints rather than the search.

11.2 Future research directions

There are a multitude of future directions that the research presented in this thesis could take. These range from directly improving on the results presented in this thesis, to applying a GE-based optimisation approach to a variety of other animation problems.

In this final section of the thesis, we suggest a variety of future work directions that a GE approach to animation could take.

Fitness function

While the presented fitness functions perform well, the simplistic weighting system could perhaps be improved upon. Although the experiment in Section 8.3, in which the fitness weights were generationally varied, did not speed up the evolution rate, this concept is certainly worthy of further examination.

Additionally, the issue of how the realism of a motion is judged could also be explored, possibly leading to the development of new, more effective fitness components.

Multi-objective evolutionary search

Motion data optimisation problems appear to be good candidates for the use of a multi-objective evolutionary algorithm, such as NSGA-II [46]. Multi-objective evolutionary algorithms perform well when a small number of objectives are involved, however, the motion data problems in this thesis have involved up to five objectives. Recently however, interest is growing in an evolutionary search method called many-objective optimisation [114] which is designed to perform well with larger numbers of objectives.

Physics-based model improvements

The manner in which the motion controller calculates the joint-torques in the physics-based models is not ideal, as instability problems are common. It may be beneficial to investigate the use of an inverse dynamics-based approach, as is common in the robotics field.

While this would increase the complexity of the motion torque calculations significantly, it would stop the stability issues caused by incorrectly set s-d coefficients. Adaptive control systems could also be investigated. These complex systems assume little about their environment and rely on feedback loops to assess the model and its environmental state at any given time. The joint-torques are then calculated based on this information.

Domain knowledge

Those grammars that did not include seed data, but implicitly included knowledge of an animal's muscles through sinusoidal functions produced results that were comparable to a real-life horse. This technique may be explored further through the implicit inclusion of other anatomical and behavioural information. The motion data generation problem could be used in future as the basis for a fuller exploration of ideal AI ratio [151].

Rate of evolution revisited

The experiments on evolution rate presented in Section 8.3 did not produce any speed-up, although they did provide some interesting results regarding generational phenotype stability. In the future, the modularly varying goals technique [98] could be more rigourously investigated.

Motion retargeting revisited

Without the s-d coefficient tuning issue, the motion retargeting experiments could be reattempted. Using more stable models, it may be possible to retarget motion data using the current system.

Physics-based Gait Transition System

The user-controllable gait and transition animation system presented in Chapter 10 is based on kinematic animation. A kinematic approach is chosen, in part, as the manner in which the transition motion data is calculated would not produce stable enough motion data for a physics-based model to move with. Additionally, the accuracy of the dynamic gait adjusters would need to be increased significantly to produce stable physics-based model motion at every Froude number.

Improvements to the torque calculation system in the physics-based model would greatly increase its tolerance of unstable data. The gait adjuster file system described in Section 10.2 could also be replaced by a dynamic motion controller. The controller could be evolved by a GE system to function in a manner similar to the gait adjuster files, while adjusting the motion in an accurate enough manner to provide stable physics-based motion. This range of improvements could allow the Kinematic Gait Transition System to be modified for use with the physics-based model.

Uneven terrain, balance and direction

The issue of uneven terrain was introduced through a short experiment in Section 8.4. The experiment was successful, although the problem itself was very narrow. Terrain traversal, balance and direction control are all determined to some extent by foot placement. Given some well-defined problem, optimal foot placements could be evolved using GE.

The problem with this is that balance and uneven terrain are not well defined, consistent problems. One could attempt to use GE to tackle these challenges by evolving a controller that could dynamically calculate the correct foot placements depending on the current state of the model and environmental information obtained from a force feedback mechanism.

Path planning

Closely linked to terrain, balance and direction is the issue of path planning. Path planning problems can range from choosing a sequence of foot placement positions, to choosing the path an animal model must take to avoid obstacles. A GE-based system similar to what is presented in this thesis could be applied to this problem.

Passive dynamics

The continuous motion retargeting system in Section 9.1.2 utilised grammars that allowed the dimensions of the model's bones themselves to evolve. This concept could be taken a step further and applied to the generation of passive dynamic models which are constructed to utilise the momentum of their swinging limbs to improve efficiency.

A passive dynamic model could be evolved using a GE-based system similar to those described in this thesis. Rather than scoring the energy efficiency of a gait motion however, the fitness function could score the energy efficiency of the model itself.

Glossary

Abduction Movement away from the body's midline, 107

Adduction Movement towards the body's midline, 107

Allele One of two or more variations of a particular gene, 326

Allometry Study of how body parts grow at different rates, 115

Amphiarthrosis A joint allowing very small movement, as found in the spine joining vertebrae, 106

Anatomical position Standing position used as a reference when describing the position of body parts in relation to each other, 335

Arity Number of arguments or operands to a function, 69

Ball and socket joint Allows flexion, extension, abduction, adduction and rotation, e.g. shoulder, hip, 108

Bone A hard rigid connective tissue which forms the endoskeleton of vertebrates, 105

Cartilage A stiff connective tissue that provides cushioning and support, 105

Caudal Located towards the tail, 336

Chromosomal crossover The exchange of genetic material between homologous chromosomes, 331

Chromosome A container for DNA within an organism's cells, 323

Codon A genetic unit of DNA or RNA comprising three nucleotides, 325

Collagen A group of proteins that make up connective tissues, 105

Conformation The shape or structure of an animal, 101

Cranial Located towards the head, 336

Cursorial Possessing limbs adapted for running, 103

Diarthrosis A joint allowing a large range of movements, as found in the shoulder, 106

Diploid Contains two sets of chromosomes, one from each parent, 323

Distal Located away from the centre (the opposite of proximal), 336

DNA Deoxyribonucleic acid: a molecule of nucleic acid which carries the instructions used in the development and functioning of all known living organisms, 322

Dorsal Located towards the back or away from the ground, 336

Extension Movement of a joint where the joint-angle increases along the sagittal plane, i.e. backward to forward, 107

Flexion Movement of a joint where the joint-angle decreases along the sagittal plane, i.e. forward to backward, 107

Frontal plane Perpendicular to the ground, divides the body into front and back halves, 336

Gene A segment of DNA that specifies some particular trait or behaviour within an organism, 323

Genetic code The genetic code is a set of rules that allow living cells to translate the information encoded in DNA or mRNA sequences into amino acid sequences, 325

Genetic recombination The rearrangement of genetic material, 330

Genome The complete set of genetic material in a cell or organism, 325

Genotype The genetic constitution of an organism, 325

Hinge joint Allows flexion and extension, e.g. elbow, knee, 108

Horizontal plane Passes through the middle of the body horizontally, dividing it into top and bottom halves, 336

Lateral Relating to the side, away from the body's midline, 336

Ligament A fibrous connective tissue that attaches bone to bone, 105

Locus The specific location of a particular gene on a chromosome, 324

Medial Relating to the middle, towards the body's midline, 336

Median plane Passes through the middle of the body vertically, dividing it into left and right halves, 336

Meiosis A process which in animals results in the formation of the gametes; ova and sperm in mammals, 331

Muscle A contractive tissue that can produce force and movement, 104

Mutation A change in the DNA sequence of a cell's genome, 332

Nucleotide Nucleotides are molecules which act as the structural units of DNA and RNA, 322

Phenotype The observable characteristics of an organism, 325

Proximal Located towards the centre (the opposite of distal), 336

Reproduction The biological process in which parent organisms produce a new individual organism, 329

RNA Ribonucleic acid: a molecule of nucleic acid central to protein synthesis, 322

Rotation Circular movement around a central point or axis, 107

Sagittal plane Passes through the body parallel to the median plane, 336

Selection A process in which genetic characteristics and environmental factors determine whether an organism survives and reproduces, 328

Synarthrosis A usually fibrous joint that allows little or no movement, as found in the skull, 106

Synovial joint See Diarthrosis, 106

Tendon A fibrous connective tissue that attaches muscle to bone, 105

Tissue A tissue is a collection of specialised cells and products of these cells, 104

Trait A distinct feature or quantifiable measurement of an organism, 323

Transcription The copying of a DNA sequence into an equivalent RNA sequence, 324

Translation A process when mRNA is decoded to produce an amino acid chain, 324

Transverse plane Passes through the body horizontally, parallel to the ground, at right angles to the median plane, 336

Appendix A

Appendix A contains additional information pertaining to Part I.

This supplementary material contains introductions, explanations and terminology relating to some of the sections in the related work portion of this thesis.

A.1 Genetics overview

Genetics is the science of variation and heredity in living organisms. Topics covered in this thesis, such as natural evolution of the horse (Section 4.1.1), breeding (Section 4.1.2) and dynamic similarity (Section 4.5) are all based around the fundamental truth that living organisms inherit the features of their parents.

A full introduction to genetics can be found in the literature [76, 82] and the following section lists the rudimentary elements, processes and terminology of genetics which relate to evolutionary computation (Section 3.2). This is followed by an overview of the evolutionary process.

A.1.1 Genetic terminology

In the following subsections, those elements of genetics which are commonly referred to in relation to evolutionary computation, are outlined. The descriptions presented here have been simplified in some cases in order to plainly communicate basic function and composition. For example, details of chemical composition and variation have been omitted as they are not relevant to the subject of evolutionary computation.

What is very relevant however, is DNA and RNA. These are fundamental elements of genetics and life in general, and this section commences with a description of both.

DNA

Deoxyribonucleic acid (DNA) is a long molecule of nucleic acid, resembling a vertically twisting ladder. DNA carries the instructions used in

the development and functioning of all known living organisms. Some segments of DNA, called genes, are used to construct cell components such as proteins and RNA (described below). Other segments of DNA control the use of genes or are present for structural purposes.

DNA comprises sequences of nucleotides which are molecules composed of a nucleobase, a sugar and between one and three phosphate groups. There are five primary types of nucleobase: cytosine (C), guanine (G), adenine (A), thymine (T) and uracil (U). In genetics these nucleobases are simply referred to as bases.

Each nucleotide in DNA contains one of four primary types of bases: C, G, A and T. DNA also contains other modified bases. The sequence of nucleotides found along the two strands of a double helix of DNA encodes information which can be read using the genetic code, described below.

RNA

Ribonucleic acid (RNA) is also a long molecule of nucleic acid which comprises nucleotides similar to DNA, but with structural and functional differences. Structurally, whilst DNA has two strands, RNA is generally single stranded and is usually shorter than DNA. In addition, the nucleotides in RNA contain the same bases of DNA except that T is replaced by U.

Functionally, RNA is fundamental to protein synthesis where messenger RNA (mRNA) carries DNA information to be be translated into proteins. This DNA information that encodes the instructions for protein construction is stored in chromosomes.

Chromosome

A chromosome is basically a container for DNA within an organism's cells. Within each chromosome, a single piece of DNA is stored compactly. The size of chromosomes varies greatly between species as the number of nucleotides in the resident DNA strand ranges between 10,000 and 1,000,000,000.

The number of chromosomes present in a cell also varies greatly between species. For example, healthy humans have 46 chromosomes whilst horses have 64. A fruit fly has 8 and a Kingfisher has 132. Most organisms are diploid meaning that they have two complete sets of chromosomes, one from each parent. These sets of chromosomes are stored in nearly every cell in an organism's body and contain an organism's genes.

Gene

A trait is defined as a distinct feature or quantifiable measurement of an organism. A gene is a segment of DNA that specifies a particular trait or behaviour within that organism. Each gene is a unit of heredity that allows traits and behaviours to be passed from parent to offspring. Traits can also be environmentally determined.

A DNA strand contains multiple individual genes. Each gene provides instructions for producing RNA or a protein which has a particular function within the organism. In humans, the number of genes stored in each chromosome ranges from 400 to 3340.

The specific location of a particular gene on a chromosome is called a locus. Each of the loci contains a gene specifying a particular trait.

The process by which information from a gene is used to realise the described trait is called gene expression and involves the transcription and translation of information encoded in DNA into the gene product.

Transcription and translation

Transcription is the copying of a DNA sequence into an equivalent RNA sequence. As DNA and RNA are structurally similar nucleic acids, certain enzymes can be used to convert back and forth between them. The stretch of DNA that is to be transcribed into an RNA molecule encodes at least one gene and is called a transcription unit. Transcription is the first step in the process of converting information from a gene into the functional gene product.

In the case of a gene which describes a protein, the transcription results in mRNA being produced. The mRNA is then used to create the protein in a process called translation, the first stage of protein biosynthesis. In translation, the mRNA is decoded to produce an amino acid chain. Processes that follow translation include post-translational modification and protein folding, which eventually produces an active protein. The translation process itself is directed by a set of rules known as a genetic code.

Genetic code

A genetic code is a set of rules that allow living cells to translate the information encoded in DNA or mRNA sequences into amino acid sequences (proteins). The genetic material (DNA or mRNA) is divided into sets of three nucleotides, called codons. A genetic code defines the

mapping between codons and amino acids, with a single codon usually specifying a single amino acid. As DNA and mRNA are constructed of a maximum of four possible nucleotide types each, there are 4^3 possible codons and multiple codons can specify the same amino acid.

Genome

The complete set of genetic material in a cell or organism is called the genome. The genome encompasses all of an organism's hereditary information including all chromosomes, their component genes and sequences of DNA present for non-coding purposes.

Whilst the genome refers to the base composition of an individual organism, the word genotype generally refers to how an individual organism's base composition is specialised or differs from other individuals of its species. Genotype is often contrasted with phenotype which is the set of observable characteristics of an organism. The phenotype arises from the interaction of the genotype with an environment.

The observable traits in the phenotype of an organism differ between individuals of the same species. This variation drives the evolutionary process and is explained by the presence of alleles.

Allele

An allele is one of two or more variations of a particular gene, found at the same locus. All organisms of a species will have the same set of genes, however, some individuals may have multiple alleles of their genes. As the DNA sequence of each allele is different, different traits may result, however, sometimes different alleles have the same resultant trait. For

example, all humans will have a gene that provides the basic instruction for eye colour, however, a green-eyed individual has a green-eye allele of that gene and a brown-eyed individual has a brown-eye allele.

Diploid organisms have a copy of each gene on each chromosome. If alleles found at a particular locus in the two corresponding chromosomes match, they are said to be homozygotes. If they are different they are said to be heterozygotes.

If the alleles at a particular locus are homozygotes, a trait encoded by both matching alleles will be observable. In the case of heterozygotes, the interaction between the disparate alleles is described as either dominant or recessive. One of the alleles is described as dominant if an observed trait is attributed to it whilst the other allele is described as recessive if its influence on a particular trait is non-existent. Often one allele is not found to be completely dominant over another. An allele is considered incomplete dominant if its trait is more predominant than the other allele, yet that other allele has some influence on the resultant trait. If that trait is found to be a near-perfect mix of the two alleles' traits, they are both considered semi-dominant. Alleles are considered co-dominant if the heterozygote's phenotype displays contributions from both alleles.

This description of alleles is important as it begins to explain how the offspring of two parents inherit a mixture of their traits. The reproductive process and the inheritance of traits is fundamental to the evolutionary process.

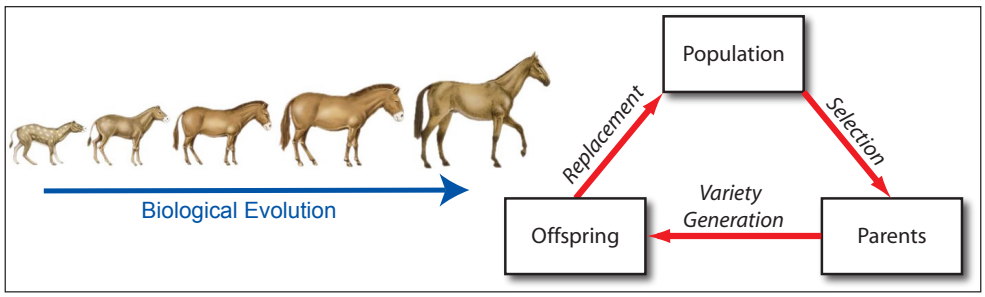


Figure A.1: The cycle of biological evolution. (Horse evolution image © National Taiwan Science Education Center [192])

A.1.2 Evolutionary processes

Evolution is the change in the inherited traits of a population of organisms though successive generations [62].

Figure A.1 presents a simplified illustration of the evolutionary engine; the process which drives biological evolution. The process occurs within a developing population of individuals in which individuals can die off or survive to produce offspring.

From a population, individuals are selected based on fitness. Through reproduction parents produce offspring. During the reproduction process, the genetic makeup of the offspring is varied through mutation and the recombination process. These offspring individuals then replace members of the current population. From this evolutionary cycle, the concepts which have led to the development of evolutionary computation are taken.

The topics which pertain directly to evolutionary computation, namely selection, reproduction, recombination and mutation are described in the following subsections.

Selection

In general, individuals selected for reproduction tend to be the fittest. The selected individuals may possess some trait or characteristic that allows them to adapt and survive in the current environment better than other members of the population. Individuals selected because of some advantageous trait may contribute a greater relative proportion of offspring to the next generation than those without that trait. If the parent individual's advantageous trait has a genetic basis, the offspring may inherit it depending on the dominance of the corresponding allele, increasing the prevalence of individuals with that trait in subsequent generations. These adaptive traits may eventually become universal to the population (or species) if selection is intense and persistent and when this occurs, the population (or species) is considered to have evolved.

Depending on the environment, selection can take place on an organism at any point in its life cycle, from the moment of conception though its time as an adult. The selection of an organism depends on selective pressures such as food, environmental conditions, predators, health and presence of mates. Regardless of selection pressures however, selection can only occur if there is variation amongst the individuals in a population. If there are no individuals in a population that can survive or take advantage of a certain selection pressure then selection will not occur.

It is this selection process that drives the evolution of a biological species. Commonly referred to as "survival of the fittest", individuals in a population are positively or negatively selected based upon their relative ability to survive and reproduce.

Reproduction

Reproduction is the biological process in which parent organisms produce a new individual organism; their offspring. All known life-forms reproduce and each individual organism is itself a product of reproduction.

Reproduction can be either asexual or sexual. In asexual reproduction, an individual organism can reproduce without the participation of another organism. The offspring individual will be genetically similar or identical to the parent as no genetic material from another individual is involved in the process. Variations in the offspring's DNA are only brought about through mutation (as described below). This provides very little variation to the genetics of offspring organisms, implying that asexual reproduction may not allow for fast adaptation to a changing environment.

Most plants have the ability to asexually reproduce. Asexual reproduction among vertebrates does occur in some reptiles and fish and on very rare occasions, in birds and sharks.

In sexual reproduction, two individuals are required, one from each sex. The offspring individuals will have a combination of the genetic material from both parents. It is usually the case that the parents are different members of the population, although only one organism is required in cases of self-fertilisation. By combining the genetic material of two individuals, as opposed to one in asexual reproduction, offspring organisms of sexual reproduction achieve greater genetic diversity. In theory, this diversity allows a species to adapt to a changing environment better than offspring created through asexual reproduction.

For species that sexually reproduce, each parent organism has a particular set of traits encoded in its own set of genes. The offspring organism inherits a single allele for each trait from each parent, ensuring that the organism has a combination of its parents' genes. The observable organism, or phenotype, will display a combination of traits from its parents depending on the dominance of each allele relative to one another.

During this reproduction process, the genetic variation of the offspring from its parents is further increased through genetic recombination.

Genetic Recombination

Genetic recombination is the rearrangement of genetic material as molecules such as DNA are broken and joined to other molecules. This rearrangement of genes promotes genetic variation, which allows natural selection to occur. In asexual reproduction, genetic variation is particularly important as it prevents the genomes in a population from accumulating irreversible deleterious mutations (permanent loss of DNA sequences).

In sexual reproduction, chromosomal crossover is a genetic recombination process that occurs during meiosis, a process that in animals results in the formation of the gametes; ova and sperm in mammals. Chromosomal crossover most often occurs as matching regions on the paired chromosomes, one from each parent, split and reconnect to the other chromosome, as illustrated in Figure A.2. This recombination shuffles the gene content between the male and female homologous chromosomes, allowing for variation in the inherited alleles that can potentially result in new beneficial allele combinations.

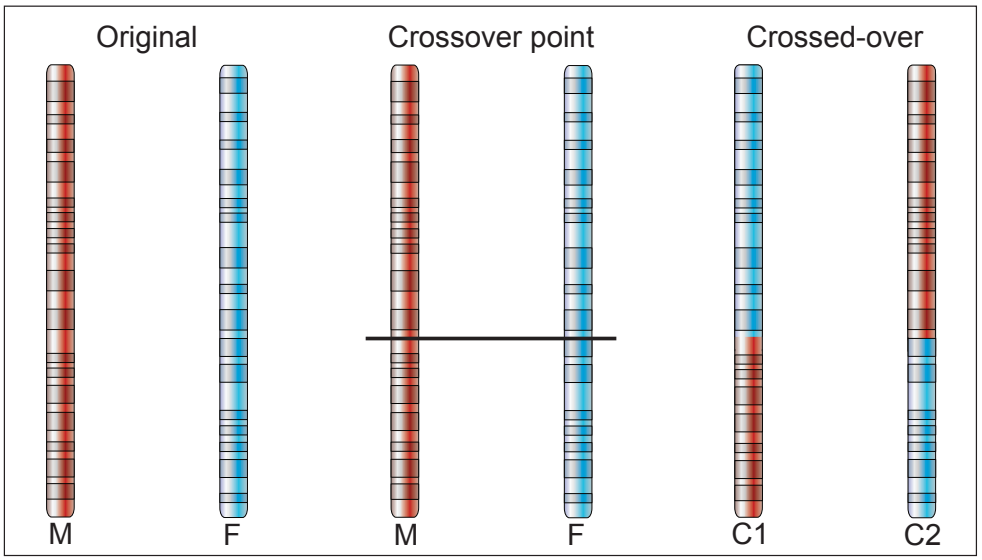


Figure A.2: Illustration of chromosomal crossover. The two chromosomes M (Male) and F (Female) are parental chromosomes which break at the crossover point and then rejoin creating new chromosomes C1 and C2, which each contain genetic material from both parents.

Occurrences of crossover imply that a selection of both parent's alleles are inherited. Without this crossover, all alleles from both parents would be inherited together.

The frequency of recombination differs between gene combinations and is described in terms of genetic distance [137], a topic which does not fall within the scope of this thesis.

Chromosomal crossover provides an element of genetic variation during sexual reproduction, and in addition to this, mutations augment the genetic variety of a population.

Mutation

A mutation is a change in the DNA sequence of an organism's genome in which the order of the nucleotides, and thus the bases, is changed.

Whilst mutations may occur due to environmental factors and viruses, this discussion focuses on errors that occur during meiosis.

As gametes are formed, mutations may have no effect, they could alter the gene product (the RNA or protein) or stop a gene from functioning altogether. Mutations occur rarely and most are deleterious or have little observable effect, however, on occasion a mutation can increase an individuals fitness and cause that individual to be favoured by natural selection. Mutations which increase fitness of a phenotype are referred to as advantageous mutations. Those which decrease fitness are called deleterious mutations. Mutations which have little or no effect on phenotypic fitness are referred to as nearly neutral and neutral mutations respectively. Some common types of mutation are show in Figure A.3.

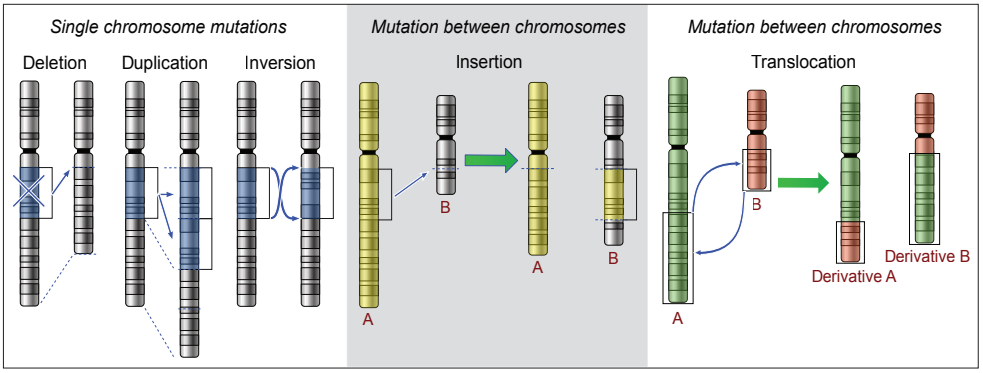


Figure A.3: Various types of chromosomal mutation. In the examples involving two chromosomes, A and B refer to distinct chromosomes.

A comprehension of the causes, types and likely locations of mutations is not prerequisite to understanding evolutionary computation techniques. The application of concepts such as mutations however, is discussed in relation to evolutionary algorithms (Sections 3.3, 3.4 and 3.5).

A.2 Horse conformation

The following are some general observations of ideal conformation that apply to most breeds of horse, adapted from [157]:

- Scapula: ideally should be 45° to the horizontal, the length of the scapula affects stride length. Horses adapted for fast running tend to have a more sloping scapula to provide greater stride length. Draught horses tend to have a more upright scapula yielding power at the expense of stride length.
- Humerus: should ideally be sloped at 60° to the horizontal for shock absorption.
- Elbow joint: the distance from the elbow to the ground should be equal to the distance from the elbow to the withers (the ridge between the scapula).
- Pastern: the hoof-pastern angle should be similar to the scapula (i.e. 45°). As with the scapula, horses adapted for fast running tend to have a more sloping pastern to give greater stride lengths whilst Draught horses tend to have a more upright pastern.
- Hoof: the ideal hoof angle is 45-50° for the forelimb and 50-55° for the the hindlimb.
- Pelvis and Hip: the movement of the pelvis and the hip should be balanced with that of the shoulder.
- Back: if the animal's back is too long it will be less able to tuck its hindquarters underneath its body during fast locomotion. If the back is too short it may lack flexibility.

The conformation of the neck can be critical to an animal's balance. The following list of neck types is adapted from [81]:

- Low-set neck: projects out from the front of the chest and is good for forward balance and moving on the forehand.
- High-set neck: rises up from the shoulders and makes collection easier (carrying more weight on the hindquarters).
- Ewe neck: dips on the top and bulges underneath with flexion at the poll joint (between neck and head).
- Bull neck: short, thick and heavy; tends to favor short strides.
- Long neck: horses with long necks can benefit from a longer stride as the long muscles of the neck help to draw forward the forelimbs, however, too much length can cause balancing issues

A.3 Terminology for anatomical location

The following is a list of terms, adapted from [77], used to describe the position of body segments of the horse (and other quadrupeds). All terms refer to an animal in a standing position (the anatomical position). Figure A.4 graphically depicts each term below with respect to a horse's body.

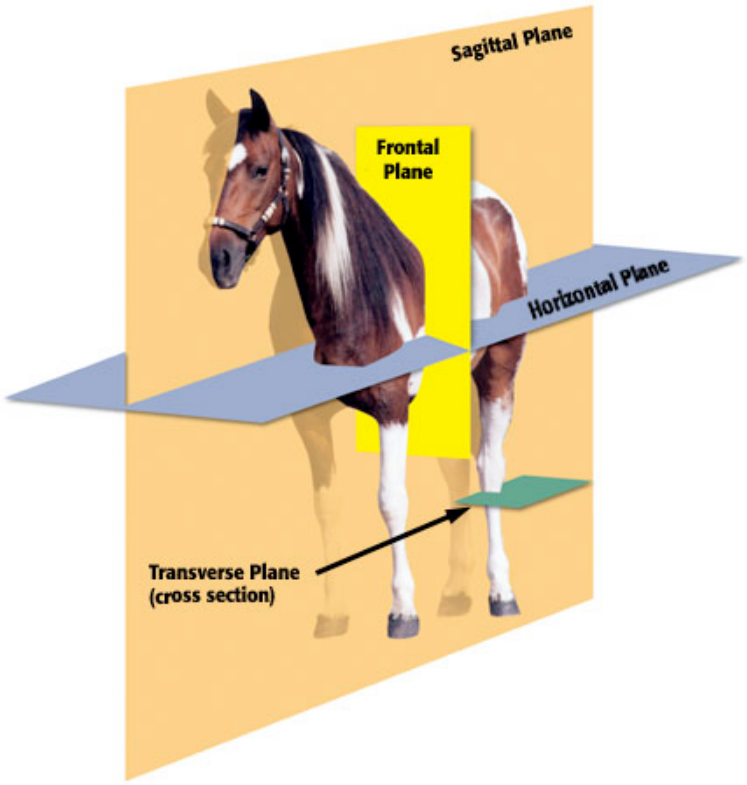


Figure A.4: Terms to describe the position and relationships of body segments of the horse. (Image taken from [193])

The following list of terms are used to describe the position of a particular part of the body relative to the body itself:

- Caudal: located towards the tail
- Cranial: located towards the head
- Distal: located away from the centre (the opposite of proximal)
- Dorsal: located towards the back or away from the ground
- Lateral: relating to the side, away from the body's midline
- Medial: relating to the middle, towards the body's midline
- Proximal: located towards the centre (the opposite of distal)

Often it is helpful to describe the positioning and movement of a body part in relation to an imaginary plane cutting through the animal's body in the anatomical position. The commonly used planes are listed below:

- Frontal plane: perpendicular to the ground, divides the body into front and back halves
- Median plane: passes through the middle of the body vertically, dividing it into left and right halves
- Sagittal plane: passes through the body parallel to the median plane
- Horizontal plane: passes through the middle of the body horizontally, dividing it into top and bottom halves
- Transverse plane: passes through the body horizontally, parallel to the ground, at right angles to the median plane

A.4 Equine gaits

The description of quadruped gaits in Section 4.4.2 was introductory. As well as its footfall sequence, a gait can be described in terms of its variations (see Section 4.4.2), symmetry and rhythm.

Symmetrical gaits are referred to as being lateral if the legs on one side of the animal move forward together. Conversely, in a diagonal gait, legs diagonally opposite from each other move forward together. Additionally, the walk is sometimes referred to as bipedal as one can observe the front and back pairs of legs separately moving in a bipedal manner (like a human).

Whilst it can be difficult for a human observing a horse in motion to visually determine a footfall sequence, each gait has an audible rhythm. The walk for instance has four footsteps at regular intervals. During the trot diagonal pairs of limbs make contact with the ground at the same time, producing only two audible beats per gait cycle. The canter and gallop have three and four beats respectively but without the regular timing displayed by the walk and the trot.

A more comprehensive list of the gaits used by horses is presented in Table A.1 which is taken from [20]. In Table A.2 the most common horse gaits are presented with specific value ranges for each of the following characteristics: speed, stride length, stride frequency, limb stance phase duration and suspension phase duration.

Table A.1: Characteristics and classifications of equine gaits.

Gait type	Gait	Gait variations	Footfall sequence	Symmetry	Rhythm (beats/stride)
Walking	Walk	collected, medium, extended	Right-hind, Right-fore Left-hind, Left-fore	Right/left bipedal	4
(Ambling)	Tölt Paso Rack Fox-trot	medium	Right-hind, Right-fore Left-hind, Left-fore	Right/left lateral	4
Running	Trot	piaffe, passage, collected, medium, extended	Right-hind & Left-fore Suspension Left-hind & Right-fore Suspension	Right/left diagonal	2
	Pace	medium, extended	Right-hind & Right-fore Suspension Left-hind & Left-fore Suspension	Right/left lateral	2
	Canter	collected, medium, extended, disunited	Trailing hind, Leading hind Trailing fore, Leading fore Suspension	Asymmetry with a phase lag between limb pairs	3
	Gallop	transverse, rotary	Trailing hind, Leading hind Trailing fore, Leading fore Suspension	Asymmetry with a phase lag between limb pairs	4

Table A.2: Value range for characteristics of equine gaits (m=metres, s=seconds).

Gait	**Speed (m/s)**	**Stride length (m)**	**Stride frequency (strides/s)**	**Limb stance phase (% stride or s)**	**Suspension phase (% stride or s)**
Walk	1.2 - 1.8	1.5 - 1.9	0.8 - 1.1	65 - 75%	0
Ambling	3.4 - 5.3	1.7 - 2.3	2.23 - 2.36	40 - 55%	0
Trot	2.8 - 14.2	1.8 - 5.9	0.9 - 2.52	26 - 53%	0 - 9%
Pace	9.1 - 16.0	4.5 - 6.3	1.8 - 2.4	0.13 - 0.138 s	0.08 - 0.094 s
Canter	2.9 - 9	1.9 - 4.6	1.6 - 2.0	0.28 - 0.30 s	0 - 0.013 s
Gallop	9 - 20	4.5 - 7.2	2.27 - 2.92	0.085 - 0.09 s	0.063 - 0.114 s

A.5 Motions of the neck and back

During the walk, the head and neck make large balancing gestures. The head tends to move up and down in a sinusoidal manner, with two similar oscillations per stride [19].

When the horse is walking in a very relaxed manner, the neck tends to be lowered. For a collected walk, the neck is raised and arched with the head near vertical. In comparison, during an extended walk, the head and neck are extended forward.

As in the walk, during the collected trot, the neck is raised and arched with the head near vertical. In the fastest racing trot, both neck and head are held very highly. The neck motion during the trot is a double oscillation movement similar to the walk, with the head reaching its highest point during the first half of each of the diagonal stance phases [19].

For the remainder of this phase, the neck lowers down to near horizontal bringing the head to its lowest point. During the suspension phase and beginning of the next stance phase, the neck is raised again.

For asymmetrical gaits such as the canter and gallop, the balancing motions required are different than that of the symmetrical gaits. During these fast running gaits, the body tends to rock upwards and downwards from the horizontal, as the hind and fore limbs impact the ground respectively.

The head and neck move in coordination with the horizontal movement of the body so the neck moves up and down in a sinusoidal manner, but with only one oscillation per stride. If the horse is moving in the col-

lected canter, the neck tends to be raised and arched with the head held at near vertical. As the canter becomes extended the neck and head extend forward also.

The gallop has neck motion similar to the canter, but as the gallop is a naturally extended gait, and the neck and head make larger forward and backward balancing gestures. Unlike the other common gaits, during the pace the neck and head do not make any balancing gestures. Instead the neck and head are held straight and high.

Appendix B

Appendix B contains additional information pertaining to Part II.

This supplementary material contains detailed explanations and actual values relating to some of the sections in the horse model construction and gait motion data portions of this thesis.

B.1 Model construction

In this section of Appendix B, supplementary information that relates to Chapter 5 is presented.

B.1.1 Measured horse data

The mass and dimensional measurements in Table B.1 are taken from [33] with a few exceptions.

The original length value for the head is measured from the head-neck attachment point to a central point on the head [33]. This measured value is extended to the length value indicated in the table to reflect the actual length of a horse's head. The length value indicated and the head's radius are estimates based on photographs and illustrations [77, 157, 81].

The radius value indicated for the neck is also an estimate based on photographs and illustrations.

The mass and length values provided for the forelimb and hindlimb are calculated from their constituent bone's data. The mass values are simply a summation. Each length value is a height value from the base of the hoof to the top of the uppermost bone in each limb. In this case the bones are positioned in a typical standing posture.

The values presented for the metacarpus and metatarsus have been reduced by the exact height of the carpal and tarsal bones respectively; these carpal and tarsal values were included in the overall measurements of their respective bones [33].

For the purpose of constructing a physics-based model, the collection of carpal and tarsal bones are not modelled. Instead a space between the

bones on either side is included. This simplification emulates the shock absorption effect of the carpal and tarsal bones.

The radii presented for both of the limbs are a roughly average bone radius value based on illustrations [77].

The trunk has been divided into two segments based on images of a horse's skeleton.

Table B.1: Horse model values (w: width, d: depth).

Segment	**Mass (kg)**	**Length (m)**	**Radius (m)**	**Angle (°)**
Head	23.1	0.5 (0.302 adjusted)	0.075	-120
Neck	26.8	0.54	n/a	n/a
Distal neck	10.344	0.22	0.055	-50
Proximal neck	16.456	0.35	0.055	-35
Forelimb	30.16	1.55701 (total height)	n/a	n/a
Scapula	11.5	0.548	0.03255	38
Humerus	8.6	0.25	0.03255	-40
Radius	6.7	0.434	0.03255	0
Carpal bones	Represented by gap	0.0191333	n/a	n/a
Metacarpus	1.59	0.267867 (0.287 adjusted)	0.03255	0
Pastern	0.73	0.135	0.03255	36
Hoof	1.04	0.075	0.0651 (w)	0
Hindlimb	31.67	1.26322 (total height)	n/a	n/a
Femur	18.6	0.36	0.03255	20
Tibia	8.3	0.434	0.03255	-35
Tarsal bones	Represented by gap	0.0235333	n/a	n/a
Metatarsus	2.84	0.329467 (0.353 adjusted)	0.03255	-9
Pastern	0.89	0.141	0.03255	25
Hoof	1.04	0.075	0.0651 (w)	0
Trunk	352.0	1.56	n/a	n/a
Thoracic	270.77	1.2	0.5208 / 0.287 (w / d)	0
Sacrum	81.23	0.36	0.2604 / 0.4823 (w / d)	0

B.1.2 Data file format

The following is a description of the horse model construction input file.

Italicised words indicate user-specified input.

BODY
Model: *any_descriptive_name*
Segments: *number*
Facing: *direction (LEFT or RIGHT)*
KDFile: *name_of_spring_constant_file*

SegmentType: TRUNK
Name: TRUNK
Bones: *number*

SBONE *name (parameters appropriate to bone)*
. . . repeat bone definitions for number of bones . . .
TRUNKSEGMENTJOINT *name articulation joins_1 joins_2 lo hi (lo1 hi1)*
. . . include above joint if number of bones greater than 1. . .
. . . repeat joint definitions for number of joints . . .
END

SegmentType: NECKHEAD
Name: NECKHEAD
Bones: *number*

BONE_TYPE name (parameters appropriate to bone)
. . . repeat bone definitions for number of bones . . .
INTRASEGMENTJOINT *name articulation axis joins_1 joins_2 lo hi (lo1 hi1)*
. . . repeat joint definitions for number of joints . . .
END

SegmentType: LIMB
LimbType: *LIMB_TYPE*
Side: *direction (LEFT or RIGHT)*
Name: *name*
Bones: *number*

BONE_TYPE name (parameters appropriate to bone)
. . . repeat bone definitions for number of bones . . .
INTRASEGMENTJOINT *name articulation axis joins_bone_1 joins_bone_2 lo hi (lo1 hi1)*
. . . repeat joint definitions for number of joints . . .
END

INTERSEGMENTJOINT *name articulation axis joins_segment_1 joins_bone_1 joins_segment_2 joins_bone_2 lo hi (lo1 hi1) x y z*
. . . repeat joint definitions for (number - 1) of segments . . .
END

END_OF_FILE

The following is an explanation of the variables found in the above file description.

number: a positive integer
direction: LEFT or RIGHT
LIMB_TYPE: FORELIMB or HINDLIMB
name (LIMB): FORELIMBLEFT or FORELIMBRIGHT or HINDLIMBLEFT or HINDLIMBRIGHT
BONE_TYPE: SBONE or RBONE or FBONE

SBONE THORACIC *mass_value length_value width_value height_value angle*

RBONE *BONE_NAME mass_value length_value radius_value angle*

FBONE *BONE_NAME mass_value radius_value angle*

name (BONE): a descriptive, unique name for this bone
name (JOINT): a descriptive, unique name for this joint
articulation: HINGE or UNIVERSAL
axis: location of rotation axis on bone-end surface: CENTRE or EDGE
joins_bone_1: the name of the first bone involved in the joint
joins_bone_2: the name of the second bone involved in the joint
joins_segment_1: the name of the first segment involved in the joint
joins_segment_2: the name of the second segment involved in the joint
lo: integer value (-π to π) defines joint's lower angular limit
hi: integer value (-π to π) defines joint's higher angular limit
lo1: integer value (-π to π) defines joint's lower angular limit (UNIVERSAL)

hi1: integer value (-π to π) defines joint's higher angular limit (UNIVERSAL)
x: offset scalar, relative to the total length of the bone, in the x direction from centre of first bone specifies joint attachment point
y: offset scalar, relative to the total height of the bone, in the y direction from centre of first bone specifies joint attachment point
z: offset scalar, relative to the total width of the bone, in the z direction from centre of first bone specifies joint attachment point

B.1.3 Data file example

The following is an example of the model input file format and values used for the horse models in this thesis.

```
BODY
Model: HORSE
Segments: 6
Facing: LEFT
KDFile: model_kd.txt

SegmentType: TRUNK
Name: TRUNK
Bones: 2

SBONE THORACIC 270.77 1.2 0.287 0.5208 0.0
SBONE SACRAL 81.23 0.36 0.4823 0.2604 0.0
TRUNKSEGMENTJOINT LUMBOSACRAL HINGE THORACIC SACRAL 0 0
END

SegmentType: NECKHEAD
Name: NECKHEAD
Bones: 3

RBONE PROXNECK 13.4 0.35 0.055 -35
RBONE DISTNECK 13.4 0.22 0.055 -50
RBONE HEAD 23.1 0.5 0.075 -120
INTRASEGMENTJOINT MIDNECK UNIVERSAL PROXNECK DISTNECK -180 180 0 0
INTRASEGMENTJOINT POLLAXIS UNIVERSAL DISTNECK HEAD -180 180 0 0
END

SegmentType: LIMB
LimbType: FORELIMB
Side: LEFT
Name: FORELIMBLEFT
Bones: 6

RBONE SCAPULA  11.5 0.548 0.03255 38.0
RBONE HUMERUS 8.6 0.25 0.03255 -40.0
RBONE RADIUS 6.7 0.434 0.03255 0.0
RBONE METACARPUS 1.59 0.287 0.03255 0.0
RBONE PASTERN 0.73 0.135 0.03255 36.0
HBONE HOOF 1.04 0.075 0.0
INTRASEGMENTJOINT SHOULDER UNIVERSAL SCAPULA HUMERUS -180 180 0 0
INTRASEGMENTJOINT ELBOW HINGE HUMERUS RADIUS -180 180
INTRASEGMENTJOINT CARPAL HINGE RADIUS METACARPUS -180 180
INTRASEGMENTJOINT FETLOCK HINGE METACARPUS PASTERN -180 180
INTRASEGMENTJOINT COFFIN HINGE PASTERN HOOF 0 0
END

SegmentType: LIMB
LimbType: FORELIMB
Side: RIGHT
Name: FORELIMBRIGHT
Bones: 6

RBONE SCAPULA  11.5 0.548 0.03255 38.0
RBONE HUMERUS 8.6 0.25 0.03255 -40.0
RBONE RADIUS 6.7 0.434 0.03255 0.0
RBONE METACARPUS 1.59 0.287 0.03255 0.0
```

```
RBONE PASTERN 0.73 0.135 0.03255 36.0
HBONE HOOF 1.04 0.075 0.0
INTRASEGMENTJOINT SHOULDER UNIVERSAL SCAPULA HUMERUS -180 180 0 0
INTRASEGMENTJOINT ELBOW HINGE HUMERUS RADIUS -180 180
INTRASEGMENTJOINT CARPAL HINGE RADIUS METACARPUS -180 180
INTRASEGMENTJOINT FETLOCK HINGE METACARPUS PASTERN -180 180
INTRASEGMENTJOINT COFFIN HINGE PASTERN HOOF 0 0
END

SegmentType: LIMB
LimbType: HINDLIMB
Side: LEFT
Name: HINDLIMBLEFT
Bones: 5

RBONE FEMUR 18.6 0.36 0.03255 20.0
RBONE TIBIA 8.3 0.434 0.03255 -35.0
RBONE METATARSUS 2.84 0.353 0.03255 -9.0
RBONE PASTERN 0.89 0.141 0.03255 25.0
HBONE HOOF 1.04 0.075 0.0
INTRASEGMENTJOINT STIFLE HINGE FEMUR TIBIA -180 180
INTRASEGMENTJOINT TARSAL HINGE TIBIA METATARSUS -180 180
INTRASEGMENTJOINT FETLOCK HINGE METATARSUS PASTERN -180 180
INTRASEGMENTJOINT COFFIN HINGE PASTERN HOOF 0 0
END

SegmentType: LIMB
LimbType: HINDLIMB
Side: RIGHT
Name: HINDLIMBRIGHT
Bones: 5

RBONE FEMUR 18.6 0.36 0.03255 20.0
RBONE TIBIA 8.3 0.434 0.03255 -35.0
RBONE METATARSUS 2.84 0.353 0.03255 -9.0
RBONE PASTERN 0.89 0.141 0.03255 25.0
HBONE HOOF 1.04 0.075 0.0
INTRASEGMENTJOINT STIFLE HINGE FEMUR TIBIA -180 180
INTRASEGMENTJOINT TARSAL HINGE TIBIA METATARSUS -180 180
INTRASEGMENTJOINT FETLOCK HINGE METATARSUS PASTERN -180 180
INTRASEGMENTJOINT COFFIN HINGE PASTERN HOOF 0 0
END

INTERSEGMENTJOINT NECK UNIVERSAL SOCKET TRUNK THORACIC NECKHEAD PROXNECK -180 180 0 0 -0.5 0.5 0.0
INTERSEGMENTJOINT LEFTFORE HINGE EDGE TRUNK THORACIC FORELIMBLEFT SCAPULA -180 180 -0.45 0.45 0.5
INTERSEGMENTJOINT RIGHTFORE HINGE EDGE TRUNK THORACIC FORELIMBRIGHT SCAPULA -180 180 -0.45 0.45 -0.5
INTERSEGMENTJOINT LEFTHIND UNIVERSAL SOCKET TRUNK SACRAL HINDLIMBLEFT FEMUR -180 180 0 0 0.45 -0.5 0.5
INTERSEGMENTJOINT RIGHTHIND UNIVERSAL SOCKET TRUNK SACRAL HINDLIMBRIGHT FEMUR -180 180 0 0 0.45 -0.5 -0.5
ENDBODY
END_OF_FILE
```

B.1.4 Physics-based model construction details

The following is a comprehensive description of the physics-based horse model construction:

Forelimb

Construction of the forelimb begins with the hoof; a block with preset mass and dimensions. The hoof is positioned sitting squarely on the ground surface.

The end point of the pastern bone is positioned above the centre of the top-right edge (lateral viewpoint with horse facing to the left) of the hoof. The joint (coffin) is a hinge with the axis of rotation aligned along that top-right edge of the hoof. Subsequent hinge joints in this limb are also aligned along this axis. The point about which this joint rotates is the centre point (the centre of the joint-end hemisphere) of the pastern. The pastern is also shifted upward a small amount to prevent any friction between the block corner and the capsule. All other limb joints avoid this friction by the addition of the calculated offsets.

The end point of the metacarpus bone is positioned above the non-connected end point of the pastern bone. A hinge joint (fetlock) rotates about the centre point (as above) of the metacarpus bone.

The end point of the radius is set above the non-connected end point of the metacarpus, with a gap between the two. This gap represents the carpal bones; a cluster of bones between the radius and the metacarpus. To simplify the simulation, these bones are not modelled. The gap inserted in their place functions similarly to the carpal bones in that it

allows some rotation and provides some shock absorbance. A hinge joint (knee) rotates about the end point of the radius.

The end point of the humerus is positioned above the non-connected end point of the radius with a hinge joint (elbow) rotating about the humerus' centre point.

The end point of the scapula is positioned above the non-connected end point of the humerus. The ODE universal joint (shoulder), a constrained ball and socket joint, rotates about the centre point of the humerus' connecting end.

With the exception of the shoulder, all of the above mentioned joints rotate about the same axis essentially constraining all movement of the bones to the sagittal plane.

In a real horse, the forelimb can move laterally as well as backwards and forwards. This lateral movement can be attributed to the scapula pulling towards and pushing out from the centre of the animal. As mentioned previously, the forelimbs are attached to the body via a sling of muscles. Although this would be possible to model, it would be nontrivial and computationally costly.

A simpler solution is to attach the top of the scapula to the trunk with a universal joint. Attaching the legs in this manner provides an additional degree of freedom with the adduction and abduction of the scapula allowing the entire leg to move laterally.

Hindlimb

As with the forelimb, construction of the hindlimb begins with the hoof. The end point of the pastern bone is positioned above the centre of the

top-right edge (lateral viewpoint with horse facing to the left) of the hoof. The joint (coffin) is a hinge with the axis of rotation aligned along that top-right edge of the hoof. The angle of the pastern is more vertical in the hindlimb than in the forelimb.

Subsequent hinge joints in this limb are also aligned along this axis. The point about which this joint rotates is the centre point of the pastern. As with the forelimb, friction and unwanted collisions are avoided by inserting a small buffer space and the use of offsets.

The end point of the metatarsus bone is positioned above the non-connected end point of the pastern bone. A hinge joint (fetlock) rotates about the centre point of the metatarsus bone. The equivalent bone in the forelimb, the metacarpus, is positioned vertically in the standing position. The metatarsus is positioned slanting slightly towards to animal's head.

The end point of the tibia is set above the non-connected end point of the metatarsus. As with the forelimb, a gap is left between the tibia and metatarsus representing the tarsal bones. These tarsal bones behave similarly to the carpal bones and again are not explicitly modelled. A hinge joint (hock) rotates about the end point of the tibia.

The end point of the femur is positioned above the non-connected end point of the tibia with a hinge joint (stifle) rotating about the femur's centre point.

In terms of bones, the hindlimb has a similar makeup to the forelimb, however, the hindlimb has no equivalent bone to the forelimb's scapula. The femur of the hindlimb attaches by a ball and socket joint to the hipbone, modelled in ODE as a universal joint.

The femur rotates about the centre point of its connecting end. A slight gap between the hip and the femur prevents unwanted collisions and friction. This gap is analogous to the lubricating fluid found in a ball and socket joint.

As with the forelimb, the series of aligned hinge joints throughout the lower limb constrain the limb's movement to the sagittal plane. The ball and socket joint (universal) at the hip allows for lateral movement.

Trunk

The trunk of the model is composed of two parts, the thoracic segment and the sacrum.

This is a simplification of a real horse skeleton, however, the behaviour of the simplified model will not differ hugely from the behaviour of the real animal's body. The vertebrae in a horse's spine do not flex a large amount from the neck to a point before the tail, therefore it is acceptable to model this segment as a single rigid body.

The thoracic segment has the forelimbs attached on either side. The width and depth of the segment is estimated from the sources named in Chapter 5.

The main flexibility found in a horse's back is at the lumbosacral joint, located close to the rear of the horse. This joint enables the horse to "tuck" its hindquarters underneath its body to a small degree. This action is useful for tight maneuvering, turning and for gaining extra power during a gallop.

The lumbosacral joint connects the thoracic segment to the sacrum which has the hindlimbs connected to its underside at the hip. This is

in contrast to the forelimbs which connect to the sides of the thoracic segment. The sacrum is therefore wider than the thoracic segment to accommodate the underside connection.

Neckhead

The positioning and movement of the neck and head has a major impact on the balance of the model.

The neck of the horse is made up of the cervical vertebrae. Like the tail, modelling many vertebrae and their connecting joints may prove difficult and computationally expensive. The neck is therefore simplified to just two articulated bones, as shown in Figure 5.1 of Section 5.1.2.

This proximal neck bone is attached to the thoracic segment via a universal joint which allows for both vertical and lateral movement of the neck. A distal neck bone is then attached to the proximal neck bone at a universal joint named the midneck joint.

At the distal end of the distal neck bone, the head is attached. In a horse, the two cervical vertebrae nearest the head are called the atlas and axis bones. These bones allow vertical and lateral rotations respectively. In this implementation, the bone representing the head is simply attached to the distal neck bone by a universal joint, emulating the freedom provided by the atlas and axis.

Positioning everything

In this implementation, there is a particular sequence of positioning and repositioning followed to ensure that all bones and segments are located correctly.

The hindlimbs are positioned first in a two-step process. The hindlimb is initially constructed at some arbitrary position, the origin for example, so that an offset can be calculated and used to position the limb at correct location underneath the sacrum.

Firstly the hooves are placed an appropriate space apart, determined by the width of the thoracic segment of the trunk. The other bones of the hindlimbs are then positioned in sequence above the hoof as previously described.

When all of the bones are connected, the global position of the centre of the hoof and the position of the attachment point on the tip of the femur are compared. The distance between the x-axis values of these two points is found (the x-axis lies parallel to the ground surface in the direction the horse is facing).

This offset is then used to reposition the hoof so that the tip of the femur lines up with the desired attachment point on the base of the sacrum.

Before the limb is joined to the sacrum segment, the trunk itself must be positioned. The limbs are positioned on the ground according to the dimensions of the trunk. The height at which the trunk is positioned is determined by the height of the topmost attachment point of the hindlimbs. Once the trunk is positioned at the appropriate height, the hindlimbs are attached to the underneath of the sacrum.

The forelimbs are positioned using the same two-step approach as the hindlimbs. Once positioned, the forelimbs can be attached to wherever the top of the scapula and thoracic segment of the trunk meet.

B.1.5 Open Dynamics Engine settings

Table B.2 lists the Open Dynamics Engine (ODE) parameter values used in all relevant experiments. An explanation of each parameter listed can be found in the ODE user manual [180].

Table B.2: ODE parameter settings.

Parameter	Value
Acceleration due to gravity	9.81
Error reduction parameter (ERP)	0.2
Constraint force mixing (CFM)	0.0001
Maximum correcting velocity	Infinity
Contact surface	0.0001
Timestep	0.0005
Contact friction	infinity
Force dependent slip in force direction 1	0.005
Force dependent slip in force direction 2	0.005
Soft ERP	0.5
Soft CFM	0.0

B.2 Gait

In this section of Appendix B, supplementary information that relates to Chapter 6 is presented.

B.2.1 Gait patterns

Gait pattern data values for the natural gaits are consistent across the literature [19]. Example values are presented in Table B.3.

Table B.3: Gait patterns: limb phase differences are shown as a percentage of a gait cycle.

Gait	Forelimb left	Forelimb right	Hindlimb left	Hindlimb right
Walk	0	50	75	25
Trot	0	49	55	6
Canter	0	80	80	50
Gallop	0	10	50	60

B.2.2 Forelimb motion data plots

The following plots are taken from [17].

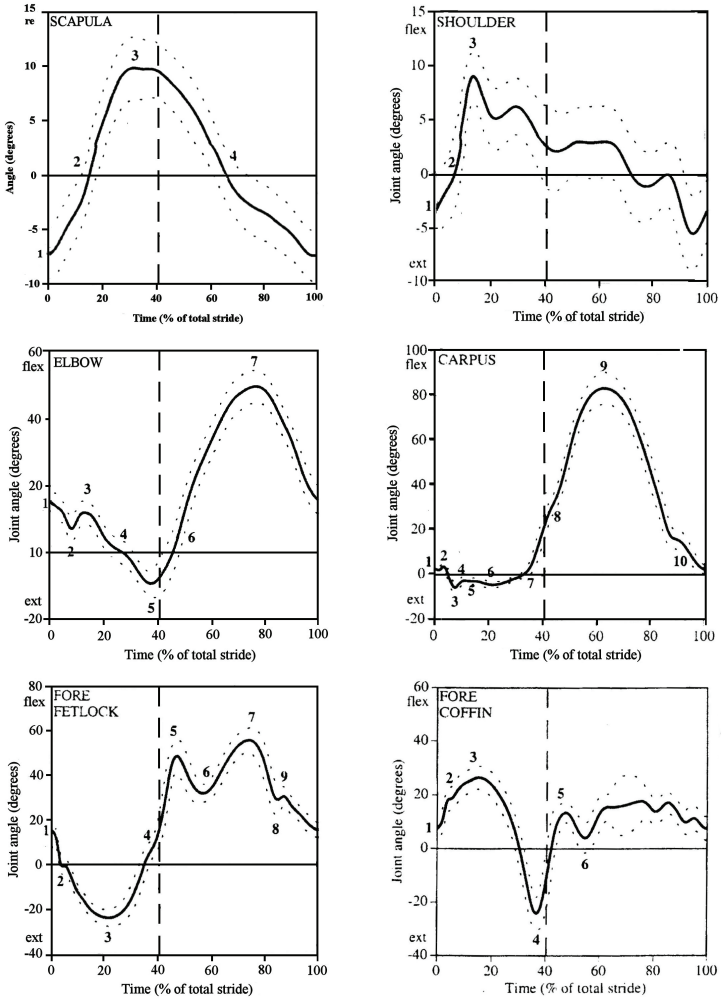


Figure B.1: Mean joint-angle-time diagrams of the forelimb of a group of horses trotting on a treadmill (4 m/s). Data are presented as mean ±SD (...). The horizontal zero line indicates the joint angle of the square standing horse. The vertical dashed line marks the transition from stance to swing (re: retraction; pro: protraction; flex: flexion; ext: extension).

B.2.3 Hindlimb motion data plots

The following plots are taken from [18].

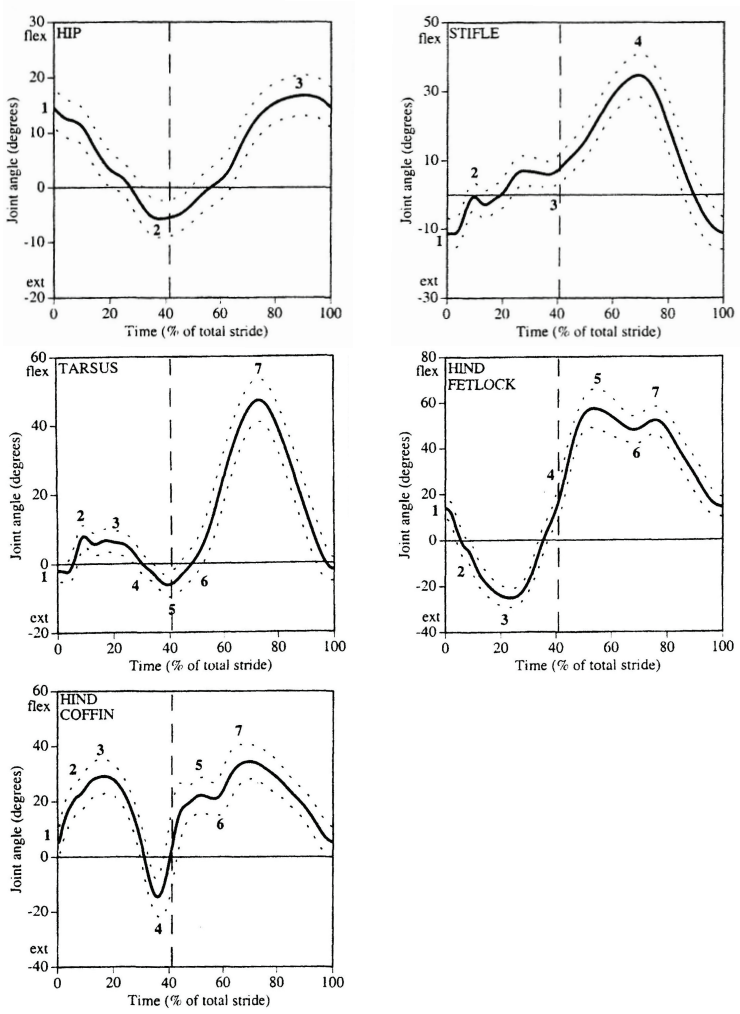


Figure B.2: Mean joint-angle-time diagrams of the hindlimb of a group of horses trotting on a treadmill (4 m/s). Data are presented as mean ±SD (...). The horizontal zero line indicates the joint angle of the square standing horse. The vertical dashed line marks the transition from stance to swing (re: retraction; pro: protraction; flex: flexion; ext: extension).

B.2.4 Forelimb motion data tables

The following tables are taken from [17] and relate to the plots presented in Appendix Section B.2.2.

Scapula

No	Time (%)	Angle (deg)	Curve
1	0	-7.3	min
2	14.9	0	zero
3	32.6	9.8	max
4	65.5	0	zero

Shoulder

No	Time (%)	Angle (deg)	Curve
1	0	-3.2	i.g.c.
2	6.3	0	zero
3	14.2	9.0	max

Elbow

No	Time (%)	Angle (deg)	Curve
1	0	15.2	i.g.c.
2	7.6	7.2	min
3	12.5	12.1	max
4	26.7	0	zero
5	37.2	-9.4	min
6	45.3	0	zero
7	76.2	49.5	max

Tarsus

No	Time (%)	Angle (deg)	Curve
1	0	2.2	i.g.c.
2	3.4	3.3	max
3	7.6	-6.0	min
4	11.2	-2.8	max
5	12.6-15.2	-3.0	plateau
6	21.4	-4.7	min
7	29.6	-1.8	bend
8	43.6	32.0	inflection
9	62.1	83.5	max
10	89.2	15.9	dip

Fore Fetlock

No	Time (%)	Angle (deg)	Curve
1	0	15.4	i.g.c.
2	4.8	-0.4	dip
3	21.5	-23.6	min
4	37.0	5.9	inflection
5	46.8	48.7	max
6	56.3	32.1	min
7	73.7	55.7	max
8	84.6	29.3	min
9	87.0	30.7	max

Fore Coffin

No	Time (%)	Angle (deg)	Curve
1	0	7.5	i.g.c.
2	4.6	18.5	dip
3	15.2	26.5	max
4	36.6	-24.1	min
5	47.5	13.4	max
6	54.6	4.0	min

Figure B.3: Tables featuring forelimb joint-angle motion values taken from [17]. Curve values indicate a distinctive feature that defines a curve's shape (i.g.c.: initial ground contact).

B.2.5 Hindlimb motion data tables

The following tables are taken [18] and relate to the plots presented in Appendix Section B.2.3.

Hip

No	Time (%)	Angle (deg)	Curve
1	0	14.5	i.g.c.
2	38.0	-5.8	min
3	90.4	16.7	max

Stifle

No	Time (%)	Angle (deg)	Curve
1	0	-11.3	i.g.c.
2	9.6	-0.5	max
3	36.4	6.0	min
4	69.1	34.6	max

Hock

No	Time (%)	Angle (deg)	Curve
1	0	-1.9	i.g.c.
2	3.4	-2.3	min
3	9.6	8.0	max
4	35.4	-3.9	zero
5	39.4	-6.2	min
6	44.2	-3.5	zero
7	73.2	47.1	max

Hind Fetlock

No	Time (%)	Angle (deg)	Curve
1	0	14.2	i.g.c.
2	7.4	-3.3	inflection
3	23.7	-25.1	min
4	37.6	6.4	inflection
5	53.9	57.4	max
6	68.1	48.0	min
7	75.8	52.3	max

Hind Coffin

No	Time (%)	Angle (deg)	Curve
1	0	5.4	i.g.c.
2	7.4	22.0	inflection
3	17.0	29.1	max
4	35.8	-14.6	min
5	51.8	22.2	max
6	56.6	21.0	min
7	69.8	34.4	max

Figure B.4: Tables featuring hindlimb joint-angle motion values taken from [18]. Curve values indicate a distinctive feature that defines a curve's shape (i.g.c.: initial ground contact).

B.3 Manual motion data generation

In this section of Appendix B, supplementary information that relates to Chapter 7 is presented.

B.3.1 MDDE data input files

The MDDE requires that the following data be supplied in simple input files:

Model a file containing model construction data, such as that discussed in Section 5.3.1 and provided in Appendix Section B.1.3

Spring-damper values a set of spring-damper coefficients that pertain to the physics-based model, as introduced in Section 5.3.3

Gait patterns the limb motion phase values for each of the natural gaits, as presented in Appendix Section B.2.1

Motion data the set of data that describes the motion of each bone in the model, for each gait

The motion data is supplied in the discrete value representation discussed in Section 6.2. For each gait, each bone's rotation over a single gait cycle must be defined by a set of 20 angular values.

Appendix C

Appendix C contains additional information pertaining to Part III.

This supplementary material contains information relating to some of the experimental portion of this thesis.

C.1 Physics-based motion optimisation

In this section of Appendix C, supplementary information that relates to Chapter 8 is presented.

C.1.1 Motion data phenotype format

The following is a description of the summation of sinusoids motion data file format. This format is used by the physics-based horse model fitness function and animation system.

The capitalised words are keywords which describe the file content:

- The GAIT keyword identifies this as a gait motion data file rather than a transition motion data file (TRANSITION).
- The SIN keyword indicates that the data in the file is in the summation of sinusoids representation. The other possible representation is piecewise, indicated by the PIECE keyword.
- The SYM keyword denotes the symmetry of the data in the file. SYM stands for symmetrical meaning that a single definition of forelimb and hindlimb motion is used for both of the forelimbs and hindlimbs. An asymmetrical file (ASYM) includes separate data for all limbs in the body, thus requiring twice as much data.

The italicised words are replaced by data in the summation of sinusoids representation. Both symmetrical and asymmetrical file formats are shown:

GAIT

SIN

SYM

scapula

humerus

radius

metacarpus

pastern (fore)

femur

tibia

metatarsus

pastern (hind)

proximal neck

distal neck

head

ENDGAIT

END_OF_FILE

GAIT

SIN

ASYM

scapula (left)

humerus (left)

radius (left)

metacarpus (left)

pastern (fore-left)

scapula (right)

humerus (right)

radius (right)

metacarpus (right)

pastern (fore-right)

femur (left)

tibia (left)

metatarsus (left)

pastern (hind-left)

femur (right)

tibia (right)

metatarsus (right)

pastern (hind-right)

proximal neck

distal neck

head

ENDGAIT

END_OF_FILE

Bibliography

[1] Michael Affenzeller and Stephan Winkler. *Genetic algorithms and genetic programming: modern concepts and practical applications*. Chapman and Hall / CRC press, 2009.

[2] Alexandros Agapitos, Julian Togelius, and Simon M. Lucas. Evolving controllers for simulated car racing using object oriented genetic programming. In *Proceedings of Genetic And Evolutionary Computation Conference*, pages 1543–1550, 2007.

[3] Aseem Agarwala, Aaron Hertzmann, David Salesin, and Steven Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics*, 23(3):584–591, August 2004.

[4] R. McN. Alexander. *Locomotion of animals*. Blackie, 1982.

[5] R. McN. Alexander. *Elastic mechanisms in animal movement*. Cambridge University Press, 1988.

[6] R. McN. Alexander. *Optima for animals*. Princeton University Press, 1996.

[7] R. McN. Alexander. Models and the scaling of energy costs for locomotion. *The Journal of Experimental Biology*, 208:1645–1652, 2005.

[8] R. McN. Alexander and G. Goldspink. *Mechanics and energetics of animal locomotion*. Chapman and Hall London, 1977.

[9] R. McN. Alexander and A. S. Jayes. A dynamic similarity hypothesis for the gaits of quadrupedal mammals. *Journal of Zoology (London)*, 201:135–152, 1983.

[10] R. McNeill Alexander. *Functional vertebrate morphology*, chapter Body support, scaling, and allometry, pages 26–37. Belknap Press of Harvard University Press, 1985.

[11] R. McNeill Alexander. *Principles of animal locomotion*. Princeton University Press, 2002.

[12] T. Amit, B.R. Gomberg, J. Milgram, and R. Shahar. Segmental inertial properties in dogs determined by magnetic resonance imaging. *The Veterinary Journal*, 182(1):94–99, 2009.

[13] Peter J. Angeline. Subtree crossover: building block engine or macromutation? In *Proceedings of the Second Annual Conference on Genetic Programming*, pages 9–17. Morgan Kaufmann, 1997.

[14] C. K. Argue and H. M. Clayton. A preliminary study of transitions between the walk and trot in dressage horses. *Acta Anatomica*, 146:179–182, 1993.

[15] C. K. Argue and H. M. Clayton. A study of transitions between the trot and canter in dressage horses. *Journal of Equine Veterinary Science*, 13(3):171–174, March 1993.

[16] Electronic Arts. Spore. `http://www.spore.com`, September 2008.

[17] W. Back, H. C. Schamhardt, H. H. C. M. Savelberg, A. J. van den Bogert, G. Bruin, W. Hartman, and A. Barneveld. How the horse moves: 1. significance of graphical representations of equine forelimb kinematics. *Equine Veterinary Journal*, 27(1):31–38, 1995.

[18] W. Back, H. C. Schamhardt, H. H. C. M. Savelberg, A. J. van den Bogert, G. Bruin, W. Hartman, and A. Barneveld. How the horse moves: 2. significance of graphical representations of equine hind limb kinematics. *Equine Veterinary Journal*, 27(1):39–45, 1995.

[19] Willem Back and Hilary M. Clayton. *Equine locomotion*. Harcourt Publishers, 2001.

[20] Willem Back and Hilary M. Clayton. *Equine locomotion*, chapter 4, page 77. Harcourt Publishers, 2001.

[21] Shaoping Bai, K. H. Low, Gerald Seet, and Teresa Zielinska. A new free gait generation for quadrupeds based on primary/secondary gait. In *Proceedings of International Conference on Robotics & Automation*, pages 1371–1376. IEEE, May 1999.

[22] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic programming: an introduction*. Morgan Kaufmann Publishers, 1998.

[23] Nils Aall Barricelli. Esempi numerici di processi di evoluzione. *Methodos*, pages 45–68, 1954.

[24] E. Barroil. *L'art équestre de haute école d'équitation: traité pratique*. J. Rothschild, 1887.

[25] A. D. Bethke. *Genetic algorithms as function optimizers*. PhD thesis, University of Michigan, Ann Arbor, 1981.

[26] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[27] B. M. Blumberg and T. A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 47–45, New York, NY, USA, 1995. ACM.

[28] Jesse Franklin Bone. *Animal anatomy and physiology*. Prentice-Hall, third edition, 1988.

[29] I. Bormann. Digitizeit 1.5 - digitizer software. `http://www.digitizeit.de`, 2003.

[30] Anthony Brabazon and Michael O'Neill. *Biologically inspired algorithms for financial modelling*. Springer, 2006.

[31] Ronald N. Bracewell. *The Fourier transform and its applications*. McGraw-Hill, 1978.

[32] C. B. Browne and F. D. Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.

[33] H. H. F. Buchner, H. H. C. M. Savelberg, H. C. Schamhardt, and A. Barneveld. Inertial properties of Dutch Warmblood horses. *Journal of Biomechanics*, 30(6):653–658, 1997.

[34] J. Byrne, J. McDermott, M. O'Neill, and A. Brabazon. An analysis of the behaviour of mutation in grammatical evolution. In *Proceedings of European Conference on Genetic Programming*, pages 14–25. Springer, 2010.

[35] Per Christiansen. Mass allometry of the appendicular skeleton in terrestrial mammals. *Journal of Morphology*, 251(2):195–209, 2002.

[36] Hilary Mary Clayton. Comparison of the stride kinematics of the collected, medium, and extended walks in horses. *American Journal of Veterinary Research*, 56(7):849–852, July 1995.

[37] Robert Cleary and Michael O'Neill. An attribute grammar decoder for the 01 Multiconstrained Knapsack Problem. *Evolutionary Computation in Combinatorial Optimization*, pages 34–45, 2005.

[38] Brian Clegg. *The man who stopped time: Eadweard Muybridge - pioneer photographer, father of the motion picture, murderer*. Sutton Publishing Ltd., July 2007.

[39] Jeff Clune, Benjamin Beckmann, Charles Ofria, and Robert Pennock. Evolving coordinated quadruped gaits with the HyperNEAT

generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing Special Section on Evolutionary Robotics*, Trondheim, Norway, May 2009. IEEE Press.

[40] Michael F. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26(2):293–302, July 1992.

[41] G. Robert Colborne, Rebecca J. Allen, Rosanna J. R. Wilson, David J. Marlin, and Samantha H. Franklin. Thoracic geometry changes during equine locomotion. *Equine and Comparative Exercise Physiology*, 3(2):53–59, 2006.

[42] Robert James Collins. *Studies in artificial evolution*. PhD thesis, University of California at Los Angeles, 1992.

[43] Nichael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 183–187. L. Erlbaum Associates Inc., 1985.

[44] W. Cui, A. Brabazon, and M. O'Neill. Dynamic trade execution strategies using grammatical evolution. In *Proceedings of European Event on Evolutionary and Natural Computation in Finance and Economics*, Istanbul, Turkey, 2010. Springer.

[45] Leandro Nunes de Castro. Fundamentals of natural computing: an overview. *Physics of Life Reviews*, 4(1):1–36, March 2007.

[46] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II.

IEEE Transactions on Evolutionary Computation, 6(2):192–197, April 2002.

[47] Ian Dempsey, Michael O'Neill, and Anthony Brabazon. *Foundations in grammatical evolution for dynamic environments*. Springer, 2009.

[48] K. G. Der, R. W. Sumner, and J. Popović. Inverse kinematics for reduced deformable models. In *ACM SIGGRAPH*, pages 1174–1179, New York, NY, USA, 2006. ACM Press.

[49] Maxwell J. Donelan and Rodger Kram. Exploring dynamic similarity in human running using simulated reduced gravity. *The Journal of Experimental Biology*, 203:2405–2415, 2000.

[50] Lenoble du Teil. *Les allures du cheval dévoilées par la méthode expérimentale*, pages 192–211. Paris and Nancy: Berger Levrault, 1893.

[51] Naoko Egi. Body mass estimates in extinct mammals from limb bone dimensions: the case of North American Hyaenodontids. *Palaeontology*, 44(3):497–528, 2001.

[52] Agoston E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer, 2003.

[53] Kenny Erleben, Jon Sporring, Knud Henriksen, and Henrik Dohlmann. *Physics-based animation*. Charles River Media, first edition, 2005.

[54] Joaquin Estremera and Pablo Gonzalez de Santos. Free gaits for quadruped robots over irregular terrain. *The International Journal of Robotics Research*, 21(2):115–130, February 2002.

[55] D. Fagan, M. Nicolau, M. O'Neill, E. Galvan-Lopez, and A. Brabazon. Investigating mapping order in πGE. In *Proceedings of IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010. IEEE Press.

[56] D. Fagan, M. O'Neill, E. Galvan-Lopez, A. Brabazon, and S. McGarraghy. An analysis of genotype-phenotype maps in grammatical evolution. *Genetic Programming*, 6010:62–73, 2010.

[57] C. T. Farley and C. R. Taylor. A mechanical trigger for the trot-gallop transition in horses. *Science*, 253(5017):306–308, 1991.

[58] L. Favreau, L. Reveret, C. Depraz, and M.-P. Cani. Animal gaits from video. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 277–286. Eurographics, 2004.

[59] Baileys Horse Feeds. Growth monitoring chart: taken from the Baileys Horse Feeds website. `http://www.baileyshorsefeeds.co.uk/studareahome/growthproblems/growthmonitorchart.htm`, March 2011.

[60] Lawrence J. Fogel, Alwin J. Owens, and Michael J. Walsh. *Artificial intelligence through simulated evolution*. Wiley (New York), 1966.

[61] Yasuhiro Fukuoko, Hiroshi Kimura, and Avis H. Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research*, 22(3-4):187–202, 2003.

[62] Douglas J. Futuyma. *Evolutionary biology*. Sunderland, 1979.

[63] E. Galvan-Lopez, D. Fagan, E. Murphy, J.M. Swafford, A. Agapitos, M. O'Neill, and A. Brabazon. Comparing the performance of the evolvable πGrammatical Evolution genotype-phenotype map to grammatical evolution in the dynamic Ms. Pac-Man environment. In *Proceedings of IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010. IEEE Press.

[64] E. Galvan-Lopez, J. M. Swafford, and M. O'Neill. Evolving a Ms. Pac-Man controller using grammatical evolution. In *Proceedings of 2nd European Event on Bio-inspired Algorithms in Games*, Istanbul, Turkey, 2010. Springer.

[65] Rockstar Games. Red Dead Redemption. `http://www.rockstargames.com/reddeadredemption`, May 2010.

[66] J. M. Guilherme Garcia and Jafferson Kamphorst Leal da Silva. Interspecific allometry of bone dimensions: a review of the theoretical models. *Physics of Life Reviews*, 3(3):188–209, 2006.

[67] Lena Mariann Garder and Mats Erling Høvin. Robot gaits evolved by combining genetic algorithms and binary hill climbing. In *Proceedings of Genetic and Evolutionary Computation Conference '06*, pages 1165–1170. ACM, July 2006.

[68] David P. Gibson, Neill W. Campbell, and Barry T. Thomas. Quadruped gait analysis using sparse motion information. In *Proceedings of 2003 International Conference on Image Processing*, volume 3, pages 333–336, 2003.

[69] Michael Girard. Interactive design of 3-D computer animated legged animal motion. In *1986 Workshop on Interactive 3D Graphics (SI3D '86)*, pages 131–150, New York, NY, USA, 1987. ACM Press.

[70] Michael Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. *ACM SIGGRAPH Computer Graphics*, 19(3):263–270, 1985.

[71] Michael Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 33–42. SIGGRAPH, ACM, 1998.

[72] Herbert Goldstein, Charles Poole, and John Safko. *Classical mechanics*. Addison-Wesley, 2002.

[73] Daoxiong Gong, Jie Yan, and Guoyu Zuo. A review of gait optimization based on evolutionary computation. *Applied Computational Intelligence and Soft Computing*, pages 1–12, 2010.

[74] D. A. Green. Growth rate in Thoroughbred yearlings and two year olds. *Equine Veterinary Journal*, 8(3):133–134, 1976.

[75] Timothy M. Griffin, Rodger Kram, Steven J. Wickler, and Donald F. Hoyt. Biomechanical and energetic determinants of the

walk-trot transition in horses. *The Journal of Experimental Biology*, 207:4215–4223, 2004.

[76] Anthony J. F. Griffiths, Jeffrey H. Miller, David T. Suzuki, Richard C. Lewontin, and William M. Gelbart. *An introduction to genetic analysis*. W.H. Freeman (New York), 2000.

[77] Pauli Grönberg. *ABC of the horse: a handbook of equine anatomy, biomechanics and conditioning*. PG-Team, first edition, 2002.

[78] Mario A. Gutiérrez, Frédéric Vexo, and Daniel Thalmann. *Stepping into virtual reality*. Springer, 2008.

[79] Michael Hardt and Oskar von Stryk. Increasing stability in dynamic gaits using numerical optimization. In *Proceedings of the 15th IFAC World Congress 2002*. Elsevier Science Ltd., 2002.

[80] Robin Harper and Alan Blair. A structure preserving crossover in grammatical evolution. *Evolutionary Computation*, 3:2537–2544, 2005.

[81] Susan E. Harris. *Horse gaits, balance and movement: the natural mechanics of movement common to all breeds*. Howell Equestrian Library. Macmillan General Reference, 1993.

[82] Daniel L. Hartl and Elizabeth W. Jones. *Essential genetics: a genomics perspective*. Jones and Bartlett (Boston), 2002.

[83] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting

to highly varied user-created morphologies. *ACM Transactions on Graphics*, 27(3):27:1–27:11, August 2008.

[84] Juliet Hedge and Don Wagoner. *Horse conformation: structure, soundness and performance*. Lyons Press, September 2004.

[85] Erik Hemberg. *An exploration of grammars in grammatical evolution*. PhD thesis, School of Computer Science and Informatics, University College Dublin, August 2010.

[86] Bonnie L. Hendricks. *International encyclopedia of horse breeds*. University of Oklahoma Press, 1995.

[87] B. Hennion, J. Villanova, J-C. Guinot, Depecker M., P. Neveu, S. Renous, and J-P. Gasc. A bio-mimetic approach to quadrupedal walking modelling. In *Proceedings of World Automation Congress*, volume 15, pages 331–338, June 2004.

[88] Hugh M. Herr and Thomas A. McMahon. A trotting horse model. *The International Journal of Robotics Research*, 19(6):566–581, June 2000.

[89] Hugh M. Herr and Thomas A. McMahon. A galloping horse model. *The International Journal of Robotics Research*, 20(1):26–37, January 2001.

[90] Milton Hildebrand. The quadrupedal gaits of vertebrates. *BioScience*, 39(11):766–775, December 1989.

[91] John H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and*

artificial intelligence. University of Michigan Press (Ann Arbor), April 1975.

[92] M. Holmström, L. E. Magnusson, and J. Philipsson. Variation in conformation of Swedish Warmblood horses and conformational characteristics of élite sport horses. *Equine Veterinary Journal*, 22(3):186–193, 1990.

[93] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata. Autonomous evolution of gaits with the Sony quadruped robot. In *Proceedings of Genetic and Evolutionary Computation*, volume 2, pages 13–17. Morgan Kaufmann, 1999.

[94] Donald F. Hoyt and C. Richard Taylor. Gait and the energetics of locomotion in horses. *Nature*, 292:239–240, July 1981.

[95] Silicon Graphics Inc. OpenGL 4.0 specification. `http://www.opengl.org/registry/doc/glspec40.core.20100311.pdf`, March 2010.

[96] Satoshi Ito, Hideo Yuasa, Zhi-wei Luo, Masami Ito, and Dai Yanagihara. A mathematical model of adaptive behavior in quadruped locomotion. *Biological Cybernetics*, 78(5):337–347, 1998.

[97] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics*, 24(3):399–407, 2005.

[98] Nadav Kashtan, Elad Noor, and Uri Alon. Varying environments can speed up evolution. *National Academy of Sciences of the USA*, 104(34):13711–13716, August 2007.

[99] A. N. Kavazis and E. A. Ott. Growth rates in Thoroughbred horses raised in Florida. *Journal of Equine Veterinary Science*, 23(8):353–357, 2003.

[100] Kazuo Kiguchi, Yukihiro Kusumoto, Keigo Watanabe, Kiyotaka Izumi, and Toshio Fukuda. Energy-optimal gait analysis of quadruped robots. *Artificial Life and Robotics*, 6(3):120–125, September 2002.

[101] Jonathan Kipling Knight and S. K. Semwal. Fast skeleton estimation from motion capture data using generalized Delogne-Kåsa method. In *Proceedings of the 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. University of West Bohemia, 2007.

[102] E. Kokkevis, D. Metaxas, and N. I. Badler. Autonomous animation and control of four-legged animals. In *Proceedings of Graphics Interface '95*, pages 10–17. Morgan Kaufmann Publishers, 1995.

[103] E. Kokkevis, D. Metaxas, and N. I. Badler. User-controlled physics-based animation for articulated figures. In *Proceedings of Computer Animation '96*, pages 16–26. IEEE, June 1996.

[104] Korg. Korg nanoKONTROL specification. `http://www.korg.co.uk/downloads/nano/support/nanoKONTROL_OM_E2.pdf`, March 2011.

[105] John R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT press, December 1992.

[106] John R. Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3):251–284, September 2010.

[107] Rodger Kram, Antoinette Domingo, and Daniel P. Ferris. Effect of reduced gravity on the preferred walk-run transition speed. *The Journal of Experimental Biology*, 200:821–826, 1997.

[108] J. Kuhnel. Rigging a horse and rider: simulating predictable and repetitive movement of the rider. Master's thesis, Texas A & M University, 2003.

[109] Marcia Kuperberg. *A guide to computer animation: for TV, games, multimedia and web.* Focal press, 2002.

[110] Shigeru Kuriyama, Yoshimi Kurihara, and Toyohisa Kaneko. Adaptive gait generation via physiological controls. In *Proceedings of Computer Animation*, pages 42–51. IEEE, 2001.

[111] W. B. Langdon and S. M. Gustafson. Genetic programming and evolvable machines: ten years of reviews. *Genetic Programming and Evolvable Machines*, 11(3):321–338, 2010.

[112] John Lasseter. Principles of traditional animation applied to 3D computer animation. *Computer Graphics*, 21(4):35–44, July 1987.

[113] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Interactive control for physically-based animation. In *Proceedings of the 2000 SIGGRAPH Conference*, pages 201–208. ACM, 2000.

[114] Miqing Li, Jinhua Zheng, Ke Li, Qizhao Yuan, and Ruimin Shen. Enhancing diversity for average ranking method in evolutionary many-objective optimization. *Parallel Problem Solving from Nature*, 6238:647–656, 2010.

[115] Jian-Nan Lin and Shin-Min Song. Modeling gait transitions of quadrupeds and their generalization with CMAC neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 32(3):177–189, August 2002.

[116] Bruce J. MacFadden. Fossil horses from "Eohippus" (Hyracotherium) to Equus: scaling, Cope's law, and the evolution of body size. *Paleobiology*, 12(4):355–369, 1986.

[117] Bruce J. MacFadden. *Fossil horses: systematics, paleobiology, and evolution of the family Equidae*. Cambridge University Press, 1992.

[118] E. J. Marey. *Animal mechanism: A treatise on terrestrial and aerial locomotion*. The International Scientific Series. Henry S. King & Co., second edition, 1874.

[119] S. A. Marsland and R. J. Lapeer. Physics-based animation of a trotting horse in a virtual environment. In *Proceedings of the 9th International Conference on Information Visualisation (IV'05)*, pages 398–403. IEEE, 2005.

[120] J. McDermott, M. O'Neill, and A. Brabazon. Interactive interpolating crossover in grammatical evolution. In *Proceedings of IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010. IEEE Press.

[121] Robert I. McKay, Hoai Nguyen Xuan, Alexander Peter Whigham, Yin Shan, and Michael O'Neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3):365–396, 2010.

[122] Michael McKenna and David Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics*, 24(4):29–38, August 1990.

[123] A. Menache. *Understanding motion capture for computer animation and video games*. Morgan Kaufmann, 2000.

[124] Mike Milne. Entertaining the future: real animators don't rotoscope, or do they? *Computer Graphics*, 32(2), May 1998.

[125] Alberto E. Minetti. Invariant aspects of human locomotion in different gravitational environments. *Acta Astronautica*, 49(3-10):191–198, 2001.

[126] R. Mistry and G. Clapworthy. Computer-based animation of a multi-legged articulated body. In *Proceedings of IEEE International Conference on Information Visualization 2000*, pages 315–317. IEEE, July 2000.

[127] Melanie Mitchell. *An introduction to genetic algorithms (complex adaptive systems)*. MIT press, February 1998.

[128] Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. Using an intermediate skeleton and inverse

kinematics for motion retargeting. *Computer Graphics Forum*, 19(3):11–19, September 2000.

[129] Natural Motion. Backbreaker. `http://www.backbreakergame.com/`, June 2010.

[130] Natural Motion. Euphoria. `http://www.naturalmotion.com/euphoria`, March 2011.

[131] E. Murphy, M. O'Neill, E. Galvan-Lopez, and A. Brabazon. Tree-adjunct grammatical evolution. In *Proceedings of IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010. IEEE Press.

[132] James E. Murphy, Hamish Carr, and Michael O'Neill. Grammatical evolution for gait retargeting. In *Proceedings of Eurographics UK Symposium on Theory & Practice of Computer Graphics*, pages 159–162, Manchester, England, June 2008. Eurographics.

[133] James E. Murphy, Hamish Carr, and Michael O'Neill. Animating horse gaits and transitions. In *Proceedings of Eurographics UK Symposium on Theory & Practice of Computer Graphics*, pages 215–222, Sheffield, England, September 2010. Eurographics.

[134] James E. Murphy, Michael O'Neill, and Hamish Carr. Exploring grammatical evolution for horse gait optimisation. In *Proceedings of the 12th European Conference on Genetic Programming*, pages 183–194, Tübingen, Germany, April 2009. Springer.

[135] James E. Murphy, Michael O'Neill, and Hamish Carr. Gait optimisation for distinct horse models using grammatical evolution. In

Proceedings of the 15th International Conference on Soft Computing MENDEL, pages 1–8, Brno, Czech Republic, June 2009.

[136] Eadweard Muybridge. *Horses and other animals in motion*. Dover Publications, 1985.

[137] Masatoshi Nei and A. K. Roychoudhury. Sampling variances of heterozygosity and genetic distance. *Genetics*, 76:379–390, February 1974.

[138] P. Neveu, J. Villanova, and J-P. Gasc. Modelisation of an unspecialised quadruped walking mammal. *Comparative Biochemistry and Physiology Part A*, 131(1):135–144, 2001.

[139] M. O'Neill and A. Brabazon. Grammatical differential evolution. In *Proceedings of International Conference on Artificial Intelligence (ICAI'06)*, pages 231–236, Las Vegas, Nevada, 2006. CSEA Press.

[140] M. O'Neill and A. Brabazon. Grammatical swarm: the generation of programs by social programming. *Natural Computing*, 5(4):443–462, 2006.

[141] M. O'Neill, A. Brabazon, M. Nicolau, S. McGarraghy, and P. Keenan. πGrammatical Evolution. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 617–629, Seattle, WA, USA, 2004. Springer.

[142] M. O'Neill, J. McDermott, J. M. Swafford, J. Byrne, E. Hemberg, E. Shotton, C. McNally, A. Brabazon, and M. Hemberg. Evolutionary design using grammatical evolution and shape grammars:

designing a shelter. *International Journal of Design Engineering*, 3(1):4–24, 2010.

[143] M. O'Neill, C. Ryan, M. Keijzer, and M. Cattolico. Crossover in grammatical evolution. *Genetic Programming and Evolvable Machines*, 4(1):67–93, 2003.

[144] Michael O'Neill and Anthony Brabazon. Recent patents on genetic programming. *Recent Patents on Computer Science*, 2:43–49, 2009.

[145] Michael O'Neill, Tony Brabazon, Conor Ryan, and J. Collins. Developing a market timing system using grammatical evolution. In *Proceedings of Genetic and Evolutionary Computation '01*, pages 1375–1381. Morgan Kaufmann, 2001.

[146] Michael O'Neill, Erik Hemberg, Jonathan Byrne, James McDermott, David Fagan, John Mark Swafford, Eoin Murphy, Conor Gilligan, Eliott Bartley, and Anthony Brabazon. GEVA: Grammatical Evolution in Java. `http://ncra.ucd.ie/geva/`, March 2011.

[147] Michael O'Neill, Erik Hemberg, Conor Gilligan, Eliott Bartley, James McDermott, and Anthony Brabazon. *GEVA - Grammatical Evolution in Java (v1.0)*. Natural Computing Research & Applications Group, University College Dublin, Ireland, December 2008.

[148] Michael O'Neill, Erik Hemberg, Conor Gilligan, Elliot Bartley, James McDermott, and Anthony Brabazon. GEVA: Grammatical Evolution in Java. *SIGEVOlution*, 3(2):17–22, 2008.

[149] Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, August 2001.

[150] Michael O'Neill and Conor Ryan. *Grammatical evolution*. Genetic Programming Series. Kluwer Academic Publishers, 2003.

[151] Michael O'Neill, Leonardo Vanneschi, Steven Gustafson, and Wolfgang Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3):339–363, 2010.

[152] Una-May O'Reilly and Martin Hemberg. Integrating generative growth and evolutionary computation for form exploration. *Genetic Programming and Evolvable Machines*, 8(2):163–186, 2007.

[153] A. Ortega, M. de la Cruz, and M. Alfonseca. Christiansen grammar evolution: grammatical evolution with semantics. *IEEE Transactions on Evolutionary Computation*, 11(1):77–90, February 2007.

[154] Grant Palmer. *Physics for game programmers*. Apress, 2005.

[155] Rick Parent. *Computer animation: algorithms and techniques*. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, 2002.

[156] Michael Raymond Pierrynowski and Victoria Galea. Enhancing the ability of gait analyses to differentiate between groups: scaling gait data to body size. *Gait and Posture*, 13(3):193–201, 2001.

[157] Sarah Pilliner, Samantha Elmhurst, and Zoe Davies. *The horse in motion*. Blackwell Publishing, 2002.

[158] Riccardo Poli, William B. Langdon, and Nicholas F. McPhee. *A field guide to genetic programming*. Free distribution, March 2008.

[159] Zoran Popović and Andrew Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH '99*, pages 11–20. SIGGRAPH, ACM, 1999.

[160] Henry D. Prange, John F. Anderson, and Hermann Rahn. Scaling of skeletal mass to body mass in birds and mammals. *The American Naturalist*, 113(1):103–122, 1979.

[161] Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. *Bezier and B-spline techniques*. Springer, 2002.

[162] Paul Rademacher. *GLUI: A GLUT-based user interface library*, version 2.0 edition, June 1999.

[163] Chantal A. Ragetly, Dominique J. Griffon, Jason E. Thomas, Ayman A. Mostafa, David J. Schaeffer, Gerald J. Pijanowski, and Elizabeth T. Hsiao-Wecksler. Noninvasive determination of body segment parameters of the hind limb in Labrador Retrievers with and without cranial cruciate ligament disease. *American Journal of Veterinary Research*, 69(9):1188–1196, September 2008.

[164] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. BigDog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress: The International Federation of Automatic Control*, pages 10822–10825, Seoul, Korea, July 2008.

[165] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics*, 25(4):349–358, 1991.

[166] Annette J. Raynor, Chow Jia Yi, Bruce Abernathy, and Quek Jin Jong. Are transitions in human gait determined by mechanical, kinetic or energetic factors? *Human Movement Science*, 21:785–805, 2002.

[167] Ingo Rechenberg. *Evolutionsstrategie - optimierung technischer systeme nach prinzipien der biologischen evolution.* PhD thesis, Department of Process Engineering, Technical University of Berlin, 1971.

[168] J. Reddin, J. McDermott, A. Brabazon, and M. O'Neill. Elevated pitch: automated grammatical evolution of short compositions. In *Proceedings of the 7th European Workshop on Evolutionary and Biologically Inspired Music, Sound, Art and Design*, Tübingen, Germany, 2009. Springer.

[169] Torsten Reil and Phil Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168, April 2002.

[170] Michael J. Reiss. *Allometry of growth and reproduction.* Cambridge University Press, 1991.

[171] C. M. B. Rodrigues and J. A. Tenreiro Machado. Fourier analysis of multi-legged periodic gaits. In *Proceedings of European Control Conference*, pages 1816–1821, September 2001.

[172] C. Rong, Q. Wang, Y. Huang, G. Xie, and L. Wang. Autonomous evolution of high-speed quadruped gaits using particle swarm optimization. In *Proceedings of 12th Annual RoboCup International Symposium*, pages 259–270. Springer, July 2009.

[173] James R. Rooney. *The lame horse*. The Russell Meerdink Company Ltd., 1998.

[174] Kisung Seo and Soohwan Hyun. Genetic programming based automatic gait generation for quadruped robots. In *Proceedings of Genetic And Evolutionary Computation Conference*, pages 293–294. ACM, 2008.

[175] Kisung Seo and Soohwan Hyun. A comparative study between genetic algorithm and genetic programming based gait generation methods for quadruped robots. *Applications of Evolutionary Computation*, 6024:352–360, 2010.

[176] Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL*. Adison-Wesley Professional, seventh edition, 2009.

[177] M. Simmons, J. Wilhelms, and A. V. Gelder. Model based reconstruction for creature animation. In *SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 139–146, New York, NY, USA, 2002. ACM Press.

[178] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*,

SIGGRAPH '94, pages 15–22, New York, NY, USA, July 1994. ACM.

[179] L. Skrba, L. Reveret, F. Hétroy, M.-P. Cani, and C. O'Sullivan. Animating quadrupeds: methods and applications. *Computer Graphics Forum*, 28(6):1541–1560, 2009.

[180] Russell Smith. *Open Dynamics Engine v0.5 user guide*, February 2006.

[181] Russell Smith. Open Dynamics Engine. `http://www.ode.org`, March 2011.

[182] Manoj Srinivasan and Andy Ruina. Computer optimization of a minimal biped model discovers walking and running. *Nature*, 439:72–75, January 2006.

[183] Manoj Srinivasan and Andy Ruina. Idealized walking and running gaits minimize work. In *Proceedings of the Royal Society A*, volume 463, pages 2429–2446. The Royal Society, July 2007.

[184] J. M. Swafford and M. O'Neill. An examination on the modularity of grammars in grammatical evolutionary design. In *Proceedings of IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010. IEEE Press.

[185] N. M. Thalmann and D. Thalmann. *Handbook of virtual humans*. John Wiley, 2004.

[186] T. Theoharis, G. Papaioannou, N. Platis, and N. M. Patrikalakis. *Graphics and visualization: principles & algorithms*. A K Peters, 2008.

[187] Frank Thomas and Ollie Johnston. *The illusion of life: Disney animation*. Hyperion Books, 1995.

[188] K. N. Thompson. Skeletal growth rates of weanling and yearling Thoroughbred horses. *Journal of Animal Science*, 73(9):2513–2517, 1995.

[189] Bill Tomlinson and Bruce Blumberg. Alphawolf: social learning, emotion and development in autonomous virtual agents. In *In Proceedings of First GSFC/JPL Workshop on Radical Agent Concepts*, pages 35–45, 2002.

[190] N. Torkos. Footprint based quadruped animation. Master's thesis, University of Toronto, 1997.

[191] N. Torkos and Michiel van de Panne. Footprint-based quadruped motion synthesis. In *Graphics Interface*, pages 151–160, 1998.

[192] Unknown. Horse evolution diagram: taken from the National Taiwan Science Education Center website. `http://activity.ntsec.gov.tw/lifeworld/english/content/images/en_evo_c6.jpg`, March 2011.

[193] Unknown. Planes of the horse. `http://www.thehorse.com/images/content/0403/horse_planes.jpg`, March 2011.

[194] Michiel van de Panne. Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, 16(2):40–49, 1996.

[195] Michiel van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, 1997.

[196] Michiel van de Panne and Alexis Lamouret. Guided optimization for balanced locomotion. In *Proceedings of Computer Animation and Simulation '95*, pages 165–177. Eurographics, Springer-Verlag, September 1995.

[197] A. J. van den Bogert. *Computer simulation of locomotion in the horse*. PhD thesis, Utrecht University, 1989.

[198] Sebastiaan van Loon. Modeling and gait control of a quadruped robot. Master's thesis, University of Twente, Netherlands, October 2005.

[199] Christopher L. Vaughan and Mark J. O'Malley. Froude and the contribution of naval architecture to our understanding of bipedal locomotion. *Gait and Posture*, 21(3):350–362, 2005.

[200] J. Villanova, J-C. Guinot, P. Neveu, and J-P. Gasc. Quadrupedal mammal locomotion dynamics 2D model. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1785–1790. IEEE/RSJ, 2000.

[201] Rodney Waschka II. *Evolutionary computer music*, chapter Composing with Genetic Algorithms: GenDash, pages 117–136. Springer, 2007.

[202] J. Wilhelms and Allen van Gelder. Combining vision and computer graphics for video motion capture. *The Visual Computer*, 19(6):360–376, 2003.

[203] Jane Wilhelms. Animals with anatomy. *Computer Graphics and Applications*, 17(3):22–30, August 2002.

[204] Jane Wilhelms and Van Gelder Allen. Anatomically based modeling. In *Proceedings of International Conference on Computer Graphics and Interactive Techniques*, pages 173–180, 1997.

[205] Andrew Witkin and Michael Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[206] Kai Xu, Xiao-ping Chen, Wei Liu, and Mary-Anne Williams. Legged robot gait locus generation based on genetic algorithms. In *Proceedings of International Symposium on Practical Cognitive Agents and Robots*, volume 213, pages 51–62. ACM, 2006.